2018

Resource Optimization in Wireless Sensor Networks for an Improved Field Coverage and Cooperative Target Tracking

Husam Sweidan
Michigan Technological University, hisweida@mtu.edu

RESOURCE OPTIMIZATION IN WIRELESS SENSOR NETWORKS FOR AN IMPROVED FIELD COVERAGE AND COOPERATIVE TARGET TRACKING

By

Husam I. Sweidan

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In Electrical Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2018

This dissertation has been approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY in Electrical Engineering.

Department of Electrical and Computer Engineering

Dissertation Advisor:	*Dr. Timothy C. Havens*
Committee Member:	*Dr. Aurenice M. Oliveira*
Committee Member:	*Dr. Laura E. Brown*
Committee Member:	*Dr. Zhaohui Wang*
Department Chair:	*Dr. Daniel R. Fuhrmann*

Dedication

This thesis is for Mom, Dad and Siblings

without whom I would neither be who I am nor would this work be what it is today.

Contents

List of Figures

List of Tables

Preface

This work presents an in-depth analysis of some of the most demanding topics in the *wireless sensor network* (WSN) research domain. Chapter 1 provides an introduction to the work presented in the dissertation and lists the main contributions of the different parts seen in the chapters thereafter. The first part mainly addresses the coverage problem for applications related to area-coverage and target tracking. Chapter 2 has been published and presented in the 2016 IEEE Congress on Evolutionary Computation (CEC) under the title *Coverage optimization in a terrain-aware wireless sensor network*. Moreover, Chapter 3 is published under the title *Sensor relocation for improved target tracking* in the IET Wireless Sensor Systems Journal, April 2018, and the work was *reproduced by permission of the Institution of Engineering & Technology*.

The second part of this work investigates the potential gains of introducing a dynamic sparse sensing algorithm into a hybrid WSN framework. The body of the topic is found in Chapter 4, and it was submitted for publication in the IEEE Transactions on Signal Processing on April 2018. Finally, a preliminary work is presented to address the destination prediction problem for mobile targets and the impact a terrain has on the prediction accuracy. This work as seen in Chapter 5 is in preparation to be submitted for publication in a journal addressing related research topics. All the work

in this dissertation was developed under the guidance of my adviser Dr. Timothy C. Havens.

The last chapter provides a conclusion of this work and suggests a set of possible future extensions for the presented material.

Acknowledgments

I would like to start by thanking my advisor Dr. Timothy C. Havens for his guidance, support and mentorship, without which this work would not be possible. It was a true privilege to work with him on several projects, some of which is not presented in this publication. I would also like to thank Dr. Zhaohui Wang for her support and guidance in the early stages of my degree.

The author would like to acknowledge the support of the Electrical and Computer Engineering Department through several research and teaching assistanships.

Finally, many thanks to my family and friends for their encouragement and advice, without which I would not get this far.

The composition of this document and the material presented in Chapter 5 was possible through the Finishing Fellowship awarded by the Graduate School at Michigan Technological University. Superior, a high performance computing infrastructure at Michigan Technological University, was used in obtaining results presented in this publication.

List of Abbreviations

AIS	Artificial Immune System
BLX	Blend Crossover
CS	Compressed Sensing
DEIF	Distributed Extended Information Filter
DKF	Distributed Kalman Filter
EM	Expectation Maximization
FFT	Fast Fourier Transform
FIFO	First In First Out
FIM	Fisher Information Matrix
FP	Frame Potential
GA	Genetic Algorithm
GDOP	Geometric Dilution of Precision
GMM	Gaussian Mixture Model
GNS	Global Node Selection
GPS	Global Positioning System
GT	Ground-Truth
HPS	High Precision Sensor
IHT	Iterative Hard Thresholding

IoT	Internet of Things
IPCA	Iterative Principle Component Analysis
IRL	Inverse Reinforcement Learning
KDE	Kernel Density Estimation
LPS	Low Precision Sensor
MDP	Markov Decision Process
MINDIST	Minimum Distant
NGA	Normalized Genetic Algorithm
NP	Non-deterministic Polynomial time
OLS	Ordinary Least Squares
OMP	Orthogonal Matching Pursuit
PCA	Principle Component Analysis
PM	Particulate Matter
PSD	Power Spectral Density
PSO	Particle Swarm Optimization
RMS	Root Mean Square
RMSE	Root Mean Square Error
ROI	Region of Interest
SNR	Signal to Noise Ratio
WSN	Wireless Sensor Network
WSS	Wide Sense Stationary

Abstract

There are various challenges that face a *wireless sensor network* (WSN) that mainly originate from the limited resources a sensor node usually has. A sensor node often relies on a battery as a power supply which, due to its limited capacity, tends to shorten the life-time of the node and the network as a whole. Other challenges arise from the limited capabilities of the sensors/actuators a node is equipped with, leading to complication like a poor coverage of the event, or limited mobility in the environment. This dissertation deals with the coverage problem as well as the limited power and capabilities of a sensor node.

In some environments, a controlled deployment of the WSN may not be attainable. In such case, the only viable option would be a random deployment over the *region of interest* (ROI), leading to a great deal of uncovered areas as well as many cutoff nodes. Three different scenarios are presented, each addressing the coverage problem for a distinct purpose. First, a multi-objective optimization is considered with the purpose of relocating the sensor nodes after the initial random deployment, through maximizing the field coverage while minimizing the cost of mobility. Simulations reveal the improvements in coverage, while maintaining the mobility cost to a minimum. In the second scenario, tracking a mobile target with a high level of accuracy is of interest. The relocation process was based on learning the spatial

mobility trends of the targets. Results show the improvement in tracking accuracy in terms of mean square position error. The last scenario involves the use of *inverse reinforcement learning* (IRL) to predict the destination of a given target. This lay the ground for future exploration of the relocation problem to achieve improved prediction accuracy. Experiments investigated the interaction between prediction accuracy and terrain severity.

The other WSN limitation is dealt with by introducing the concept of sparse sensing to schedule the measurements of sensor nodes. A hybrid WSN setup of low and high precision nodes is examined. Simulations showed that the greedy algorithm used for scheduling the nodes, realized a network that is more resilient to individual node failure. Moreover, the use of more affordable nodes stroke a better trade-off between deployment feasibility and precision.

Chapter 1

Introduction

The advancements in electronics and communications made it possible to design and build sensory nodes that are compact, power efficient and economically feasible. This paves the road to the utilization of sensory networks in many applications to monitor and record phenomena or a certain activity. To mention few, they can be used for: (1) environmental monitoring of air or water quality, (2) measuring the soil moisture levels in farms, (3) tracking and predicting enemy movement in a war zone. Moreover, the current trends aim at an even more substantial use of sensory networks. The concept of the Internet of Things (IoT) is becoming more predominant in our daily lives. As shown by more and more devices that are equipped to collect and communicate data, like toasters to the vehicles we drive. Sensory networks can be either wired, wireless or a hybrid of both. Wired sensing nodes can be advantageous in terms of having

a more stable power source and communication medium, while wireless nodes offer more flexibility in network's spatial deployment. This work mainly discusses issues and challenges related to the use of wireless sensor networks (WSN) or the use of wireless nodes to augment a wired sensor network which forms a hybrid network.

Many challenges arise in using a WSN for a given application. Wireless sensor nodes typically have limited resources, namely, battery life, processing power, and sensors/actuators capabilities, which leads to: (1) Limited coverage of the event under observation, and (2) a shortened life-time, leading to node failures and hence compromising the overall network reliability. This dissertation tackles those challenges by optimizing the use of the available network resources. The sections below present a brief description on how those challenges were addressed, with a more comprehensive discussion provided in the following chapters.

1.1 Coverage Optimization in a WSN

In many scenarios a planned deployment of a WSN can be complicated and unfeasible due to the hostility of the environment. A couple examples of such scenarios would be a sensory network for monitoring an active volcano, or an infiltrator detection and tracking in a war zone. In such cases the only viable option would be an airdrop deployment of the network in the region of interest (ROI). This creates a randomly

distributed network that suffers from coverage holes and connectivity break-offs. Coverage holes can lead to a poor depiction of the actual event of interest, and in case of tracking applications an imprecise estimations or utmost a lost tracking. Moreover, having a group of the sensing nodes disconnected due to the random deployment oversight the full potential of the network.

In chapters 2, 3 and 5, the problem of field coverage is addressed from different angles and for different applications. A common theme among those chapters is the study of the impact a terrain has over the proposed methods and solutions. In chapter 2 the problem of sensor relocation after a random initial deployment is tackled. As mentioned earlier, this is of importance due its potential of mending the coverage holes' problem. The proposed WSN has nodes that are capable of moving across the ROI, but since the the nodes rely on a limited power source, the cost of mobility is high. A multi-objective optimization problem is presented with the purpose of maximizing the area covered by the network, while minimizing the cost of mobility. Mobility cost was introduced to the objective function through both the traveled distance and the severity of the terrain. Due to its relation to the set coverage problem this optimization is considered to be NP-complete, hence the use of evolutionary computation algorithms was considered [14]. Three algorithms were used for this purpose: the Artificial Immune System (AIS) algorithm, the Genetic Algorithm (GA), and the Particle Swarm Optimization (PSO) algorithm . It was shown in the results that both the AIS and GA outperformed the PSO especially

for less dense networks, while the PSO offered a decent performance with a lower execution time [1].

Chapter 3 explores the potential gain of sensor node relocation on the accuracy of target tracking. The nodes in the proposed WSN are assumed to have the ability to be mobile. The presented system is initialized with randomly distributed nodes with the purpose of tracking any moving targets within the field. From this initial state, a database of target location estimates is formed with the intention of using it to learn the mobility trends of the targets of interest. The kernel density estimation (KDE) algorithm is used to estimate a spatial distribution of the previously recorded location estimates, where it is used to establish a ROI the reflects the preferences of the mobile targets. Having established the ROI, the next phase would be relocating a set of sensor nodes to this region, where the nodes are selected based on their distance from the ROI centroid. Several methods are tested to optimize the positioning of the relocated nodes inside the ROI with the intent of achieving a better target position estimates. The first method is based on the geometric dilution of precision (GDOP) metric adapted for our 2D scenario [44]. The GDOP is a dimensionless measure usually used in the satellite navigation domain as an indicate of positioning precision. The second method depends on the K-coverage measure which essentially makes sure that a given point in the field is covered by at least K sensors. Finally, a simple relocation to a uniformly distributed random locations inside the ROI. The K-coverage offered the best performance prominently for sensor nodes with short to medium detection

range [79].

Path and destination prediction for mobile targets has a significant potential in WSNs. For instance, instead of relocating nodes in the whole ROI, it would be more efficient to move a smaller set to cover an area where a target is expected to be. Chapter 5 investigates the use of *inverse reinforcement learning* (IRL) to predict the destination of a moving target based on observing a portion of its trajectory. Targets with different capabilities for traversing a given terrain are considered, where terrain severity is used to generate a set of features with the purpose of learning the preferences and capabilities of the target under investigation. For a varying observed trajectory lengths, the accuracy of prediction is used as a performance measure.

A brief description of the contributions presented in Chapters 2, 3 and 5 are:

† Chapter 2: The use of evolutionary computation to address a multi-objective problem considering a trade-off between coverage rate and mobility cost.

† Chapter 3: Introduce an algorithm for relocating sensor nodes to a region of interest that is deduced based on targets mobility trends, with the purpose of improving the tracking accuracy.

† Chapter 5: Investigating the impact of field's terrain on the accuray of destination prediction.

1.2 Sparse Sensing and Measurement Scheduling

Extending the life-time of a WSN requires a good management of the power resources at the individual node level. It is often common to invest more into optimizing the power consumed by data processing and communications, and less into sensing and data collection. An optimized power management system requires addressing all three areas. In chapter 4 the concept of sparse sensing is introduced to a hybrid network. The hybrid network consists of a sparsely distributed high precision sensor (HPS) nodes, as well as a larger group of low precision sensor (LPS) nodes that are more economically feasible. The goal was to activate a small set of the nodes to measure the phenomena of interest while retaining sufficient information to reconstruct the data of the inactive nodes. Three main methods were investigated to schedule the nodes for measurements: (1) A simple selection of uniformly spaced nodes, and two greedy algorithms with the first based on the (2) frame potential (FP) measure [61, 71], and the other on the (3) correlation measure [72]. The main performance figure of merit was the accuracy of the reconstruction based on the root mean square error (RMSE). Both the uniform and the FP methods offered a superior performance over the correlation approach, with a small edge for the uniform method. Even though the uniform scheduling offered the best reconstruction performance, simulations showed that it is less resilient than the FP based greedy algorithm. More over, experiments showed that augmenting a sparsely distributed network of HPS nodes with large

number of LPS nodes, offered a better balance between reconstruction performance and network deployment feasibility, as compared with an all HPS nodes network. The contribution of as compared to related work in the literature is as follows:

† Introducing the sparse sensing concept into the unique hybrid of LPS and HPS nodes for measurement scheduling, revealing an improvement in reconstruction accuracy while preserving a lower deployment cost.

† Exploring the resilience of the greedy measurement-scheduling algorithms to sensor node failures.

1.3 Publications

The work presented in this dissertation is mainly based on the following publications:

† Chapter: 5

– H.I. Sweidan and T.C. Havens., *"Destination Prediction of Terrain-Aware Mobile Agents Via Inverse Reinforcement Learning,"* In preparation.

† Chapter 4

– H.I. Sweidan and T.C. Havens., *"Dynamic Greedy Scheduling for Sparse Sensing in Hybrid Sensor Networks,"* Submitted. IEEE Trans. Geoscience

and Remote Sensing.

† Chapter 3

- H.I. Sweidan and T.C. Havens., *"Sensor relocation for improved target tracking,"* IET Wireless Sensor Systems, 2018.

† Chapter 2

- H.I. Sweidan and T. C. Havens., *"Coverage optimization in a terrain-aware wireless sensor network,"* 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, pp. 3687-3694. 2016.

Chapter 2

Coverage Optimization in a Terrain-Aware Wireless Sensor Network

2.1 Introduction

Regardless of the application for which a *wireless sensor network* (WSN) is used, coverage is a critical factor that directly impacts the quality of service. There are two main types of WSN coverage addressed in the literature: i) area coverage, where the interest is in maximizing the covered area in a given *region of interest* (ROI)

[2, 3, 4, 5, 6, 7], and ii) target coverage, where covering static or mobile targets is of essence [8, 9, 10, 11, 12].

For large WSNs, controlled deployment of the sensing nodes not only can be complex, but it is also not practical especially in hostile environments. In such case, random deployment is usually the method of choice. However, random deployment can cause coverage holes, which degrades the effectiveness of the WSN [13]. Therefore, it is necessary to relocate the sensing nodes after the first deployment to mend the coverage holes. In most cases power is a very limited resource in a WSN, where if mismanaged, it would drastically shorten the lifetime of the network. Sensor relocation requires mobility, which is an energy exhausting operation. The amount of energy spent on mobility can be directly associated to: i) traveled distance, and ii) severity of the terrain. Hence, a relocation algorithm is required to maximize the covered area, while keeping the energy spent on mobility at a minimum. This problem is related to the set coverage problems and is considered to be NP-complete [14]. Accordingly, evolutionary computation techniques would be a reasonable choice to investigate this problem.

Table 2.1 provide a description of the important notations used in the following sections. The rest of the chapter is organized as follows. Section 2.2 provides a literature survey for similar problems. The problem structure and the used methods are presented in Section 2.3. Section 2.4 briefly describes the algorithms used in this

Table 2.1
Notations

Term	Definition	Term	Definition
N_x, N_y	Field dimensions	N	Number of sensors
R_s	Sensing range	R_c	Communication range
r_e	Range error	Pr_{th}	Probability threshold
a, b	Covarience model parameters	α	optimization trade-off parameter
G_{th}	Gradient threshold	P_s	Population size
P_r	Replication rate	P_c	Clonal percentage
P_h	Hypermutation rate	P_m	Mutation rate
$g_{\max}$	Max. number of generations	$w_{\min}$, $w_{\max}$	Inertia limits
c_1, c_2	PSO design parameters	$Itr_{\max}$	Max. number of iterations

work. Simulation results and discussion are found in Section 2.5. Finally, Section 2.6 concludes the work.

2.2 Related Work

Coverage optimization problems for WSNs has been tackled many times in the literature. For limited mobility nodes, an *artificial immune system* (AIS) algorithm was investigated to maximize the coverage area [2], where the traveled distance was utilized as a measure for mobility cost. Yoon and Kim introduced a novel normalization method to the *genetic algorithm* (GA), which reduced the redundancy in the solution space resulting in a better performance [3]. Particle swarm optimization has also been examined to resolve the coverage problem. A combination of Voroni diagram

and a two-phase *particle swarm optimization* (PSO) were used, where the first phase maximizes the coverage area and the second phase minimizes the traveled distance [5]. With regard to target coverage, Zhuofan *et al.* investigated reducing the mobility cost when solving for both fixed target coverage and network connectivity problems [9]. Mobile target coverage in a vehicular ad hoc sensor network was presented in [11]. This chapter considers the problem of coverage area optimization for sensor relocation after initial deployment. Since energy expenditure of mobility in a WSN is immense, it was penalized by searching for a minimum relocation distance traveled through a terrain with modest severity, where terrain severity is represented by its gradient. Three algorithms are used to inspect this problem, the AIS algorithm, the PSO, and the normalized GA. Although the impact of terrain on sensor deployment has been studied in the literature [15, 16, 17], its impact on the relocation problem requires further investigation. The contribution of the work presented in this chapter reside in the use/comparison of evolutionary computation algorithms to address,

† A multi-objective problem considering a trade-off between coverage rate and mobility cost.

† Study the impact of terrain severity on the mobility of sensor nodes.

2.3 Problem Structure

2.3.1 ROI Coverage

A square ROI is considered in this work, where a $N_x \times N_y$ grid is superimposed over it. Each grid element is of 1×1 size, where a point on the grid is expressed as $P(x, y)$. N sensing nodes are uniformly distributed in the ROI as an initial deployment, such that $s_n(x_n, y_n)$ is the nth sensor node in the grid, and for simplicity is expressed as s_n. The sensing field of each sensor is considered to be a circle with a radius R. The probabilistic sensing model is used, where a sensor's detection uncertainty is accounted for by [2, 18]

$$Pr_{\mathrm{x,y}}(s_n) = \begin{cases} 1 & R - r_e \geq d(s_n, P), \\ \exp^{-a\lambda^b} & R - r_e < d(s_n, P) < R + r_e, \\ 0 & R + r_e \leq d(s_n, P), \end{cases} \tag{2.1}$$

where r_e is the error in the sensing range—see Fig. 2.1. The distance between sensor s_n and point $P(x, y)$ is represented by $d(s_n, P)$. The parameters a and b are for measuring the detection probability in the uncertainty region (i.e., $R - r_e < d(s_n, P) < R + r_e$), and $\lambda = d(s_n, P) - (R - r_e)$. It is clear that the detection

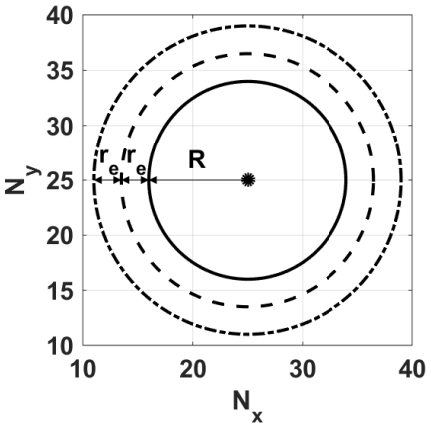


Figure 2.1: Sensing coverage model.

probability $Pr_{\mathrm{x,y}}(s_n)$ decays exponentially with distance in the uncertainty region.

The first objective of this problem is to maximize the area covered by the N sensors. It is first required to calculate the coverage rate,

$$C_{\mathrm{rate}} = \frac{C_{\mathrm{area}}}{A_{\mathrm{total}}}, \tag{2.2}$$

where $A_{\mathrm{total}} = N_x \times N_y$ is the total area of the ROI. Also, the coverage area C_{area} is given by,

$$C_{\mathrm{area}} = \sum_{x=0}^{N_x} \sum_{y=0}^{N_y} Pr_{\mathrm{x,y}}(\mathcal{S}), \quad \forall Pr_{\mathrm{x,y}}(\mathcal{S}) \geq Pr_{\mathrm{th}}, \tag{2.3}$$

where $\mathcal{S} = \{s_1, s_2, ..., s_N\}$ is the set of all sensors, and $Pr_{\mathrm{x,y}}(\mathcal{S}) = 1 - \prod_{n=1}^{N}(1 - Pr_{\mathrm{x,y}}(s_n))$ is the coverage probability of point $P(x, y)$ considering the sensors in $\mathcal{S}$. If $Pr_{\mathrm{x,y}}(\mathcal{S})$ exceeds the threshold probability Pr_{th}, then its value is carried on for the evaluation of (2.3). Now that C_{rate} is attained, coverage optimization can be carried out by minimizing the rate of the uncovered area, $\bar{C}_{\mathrm{rate}} = 1 - C_{\mathrm{rate}}$.

2.3.2 Mobility Cost

2.3.2.1 Traveling Distance

This work examines two traveling methods. The first is along a two-point path where the first point is the one of initial deployment and the second is the destination. The other method is carried along a path computed using the A-star algorithm to avoid severe terrains—details in Section 2.3.2.2. Obviously the A-star path is at least as long as the two-point path. The traveling distance of relocating the sensors in $\mathcal{S}$ is assessed by the *root-mean-square* (RMS) of their individual traveling distances,

$$d_{\text{RMS}} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} d(s_n, P_{n,f})^2}, \tag{2.4}$$

where $P_{n,f}$ is the final point of the nth sensor relocation path. The distance $d(s_n, P_{n,f})$ is expressed as,

$$d(s_n, P_{n,f}) = \sum_{m=2}^{M_n} \sqrt{(x_m - x_{m-1})^2 + (y_m - y_{m-1})^2}, \tag{2.5}$$

where M_n is the number of points on the nth sensor relocation path, with $M_n = 2$ for the two-point path, and $M_n \geq 2$ for the A-star path.

2.3.2.2 Terrain Severity

Here, the severity of the terrain is mainly quantified by its steepness. For instance, a steep downhill or uphill terrain is considered more difficult to traverse, and hence not preferable for sensors to pass through. Steepness is computed by evaluating the gradient of the terrain and then taking its absolute value. Therefore, the severity of the nth sensor relocation path is given by

$$Sev(s_n, P_{n,f}) = \frac{\sum_{m=1}^{M_n} |G(P_{n,m})|}{M_n}, \ \forall M_n \geq 2, \tag{2.6}$$

where $P_{n,m}$ is the mth point on the nth sensor relocation path, and $G(P_{n,m})$ is the gradient at that point. The overall relocation severity can be measured as

$$Sev_{\text{RMS}} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} Sev(s_n, P_{n,f})^2}, \tag{2.7}$$

where Sev_{RMS} is the RMS of the relocation severity for the sensors in $\mathcal{S}$. When considering the A-star method, parts of the terrain may be perceived as an obstacle based on the value of gradient. A point $P(x, y)$ in the grid is considered to be an obstacle if

$$|G'(P(x, y))| \geq G_{\text{th}}, \tag{2.8}$$

where $G'(P(x,y))$ is the normalized gradient at point $P(x,y)$, and $G_{\text{th}} \in [0,1]$ is a threshold parameter for the gradient.

2.3.3 Objective Function

For the problem at hand, maximizing the coverage while keeping the mobility cost at a minimum is of interest. Accordingly, the objective function can be expressed as,

$$\min \left\{ F = \alpha \bar{C}_{\text{rate}} + (1-\alpha) \frac{d_{\text{RMS}}\ Sev_{\text{RMS}}}{R_c} \right\}, \tag{2.9}$$

where $\alpha \in [0,1]$ is a design parameter that allows the user to select the trade-off between the coverage and mobility costs. The parameter R_c is the sensor's communication range, and it is used in the objective function to count for the network connectivity.

2.4 Algorithms

The algorithms used to assess the problem at hand are either evolutionary (GA and AIS) or related to the evolutionary techniques (PSO), where they rely on a population of solutions in the search for optimality. The structure of a member in a population is illustrated in Fig. 2.2.

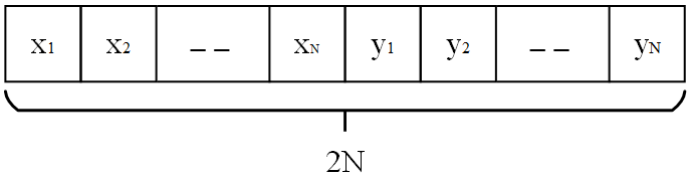


Figure 2.2: The structure of a population member.

2.4.1 Artificial Immune System Algorithm

The following provides some details regarding the AIS algorithm, outlined in Algorithm 1 [2, 19].

Algorithm 1 AIS Algorithm

$PoP \leftarrow$ Initialize the population.
$n_g = 0$ (generation counter).
while $n_g \leq g_{\max}$ **do**
  $F(\mathcal{S}, PoP) \leftarrow$ Antibody fitness evaluation.
  $PoP_{\text{sel}} \leftarrow$ Fitness proportionate selection.
  $PoP_{\text{rep}} \leftarrow$ Replicate best $P_r \times P_s$ antibodies.
  $PoP_{\text{clo}} \leftarrow$ Apply clonal proliferation over PoP_{rep}.
  $PoP_{\text{hyp}} \leftarrow$ Hypermutation over PoP_{clo}.
  $PoP_{\text{tot}} \leftarrow [PoP_{\text{rep}}; PoP_{\text{hyp}}]$.
  $PoP_{\text{child}} \leftarrow$ Mutation over PoP_{tot}.
  $PoP \leftarrow$ Select Best P_s antibodies out of PoP_{child}.
  $n_g = n_g + 1$
end while
$\mathcal{S}_{\text{final}} \leftarrow$ Antibody with the minimum fitness.
return $\mathcal{S}_{\text{final}}$

2.4.1.1 Fitness Proportionate Selection

The selection process is carried out by first normalizing the fitness value of each antibody in the population,

$$F_{\text{norm}} = \frac{F(\mathcal{S}, PoP)}{\sum_{l=1}^{P_s} F(\mathcal{S}, PoP(l))}, \tag{2.10}$$

where $F_{\text{norm}} \in [0, 1]$ is a vector containing the normalized fitness values, and $F(\mathcal{S}, PoP)$ computes the fitness of each antibody in the population. $PoP(l)$ is the lth antibody in the population, and P_s is the population size. The next step is to calculate the cumulative sum of the values in F_{norm}, and then select the first antibody with a cumulative fitness that exceeds a random number $r \sim U[0, 1]$. The selection based on the random number r is repeated P_s times and the resulting antibodies are stored in PoP_{sel}.

2.4.1.2 Replication

The P_s antibodies chosen by the fitness proportionate selection are sorted in ascending order according to their fitness values, and the first $P_r \times P_s$ antibodies are kept in PoP_{rep}, where $P_r \in [0, 1]$ is the replication rate.

2.4.1.3 Clonal Proliferation

A random number $r \sim U[0,1]$ is assigned to each antibody in PoP_{rep}. For the case where the clonal percentage P_c is greater than r, that antibody is placed in PoP_{clo}.

2.4.1.4 Hypermutation

Every antibody that joined the clonal proliferation goes under the hypermutaion process. For an antibody $\mathcal{A} = \{J_1 J_2 ... J_k ... J_{2N}\}$, any given gene could be chosen for hypermutation depending on the rate P_h. Assume that gene J_k was chosen for hypermutation, another gene J_i is then randomly selected to join the operation. The new gene $J'_k = (1-\Gamma)J_k + \Gamma J_i$, for $\Gamma \sim U[0,1]$, and is stored in PoP_{hyp}.

2.4.1.5 Mutation

Similar to the hypermutation process, mutation promotes exploration through diversifying the antibodies. With a rate P_m much lower than P_h, mutation is applied over the antibodies in PoP_{rep} and PoP_{hyp}. For an antibody $\mathcal{A} = \{J_1 J_2 ... J_k ... J_{2N}\}$, let J_k be a gene selected with a rate P_m to undergo mutation. A pairing gene J_i is randomly

selected from $\mathcal{A}$ to join the process. The new mutated genes are expressed as

$$J'_k = (1 - \Gamma)J_k + \Gamma J_i, \tag{2.11}$$

$$J'_i = (1 - \Gamma)J_i + \Gamma J_k, \tag{2.12}$$

where $\Gamma \sim U[0, 1]$.

2.4.2 Normalized Genetic Algorithm (NGA)

In this type of problem, redundancy imposes itself in the solution (phenotype) space. For instance, let us express a solution (chromosome) $\mathcal{A}_i$ in the phenotype space as a set of 2D location pairs for the sensors in $\mathcal{S}$; $\mathcal{A}_i = \{(x_1, y_1)(x_2, y_2)...(x_N, y_N)\}$. Another solution $\mathcal{A}_j$ might exist with a set of pairs that are only a rearrangement of the ones in $\mathcal{A}_i$. This will cause similar results in terms of coverage, but not necessarily in traveling distance or path severity. The normalized GA introduced by Yoon and Kim, outlined in Algorithm 2, addresses this issue [3]. The following provides some details regarding the normalization, crossover, and mutation operations used in this algorithm.

Algorithm 2 Normalized Genetic Algorithm (NGA)

$PoP \leftarrow$ Initialize the population.
$n_g = 0$ (generation counter).
while $n_g \leq g_{\max}$ **do**
  $\{\text{Parents}_1, \text{Parents}_2\} \leftarrow$ Random pairing.
  $\text{Parents}_2' \leftarrow$ MINDIST normalization on Parents_2.
  OffSpring $\leftarrow$ BLX-0.5 crossover.
  $\text{OffSpring}' \leftarrow$ Gaussian mutation over OffSpring.
  $PoP_{\text{tot}} \leftarrow [PoP; \text{OffSpring}']$
  $F(PoP_{\text{tot}}, \mathcal{S}) \leftarrow$ Chromosome fitness evaluation.
  $PoP \leftarrow$ Select best P_s chromosome out of PoP_{tot}.
  $n_g = n_g + 1$
end while
$\mathcal{S}_{\text{final}} \leftarrow$ Chromosome with the minimum fitness.
return $\mathcal{S}_{\text{final}}$

2.4.2.1 Minimum Distance (MINDIST) Normalization

Before starting the MINDIST normalization, it is necessary to randomly pair the P_s chromosomes into $P_s/2$ parent pairs. For illustration, let the chromosomes be represented as a set of 2D location pairs, with $\mathcal{A}_{l,1} = \{(x_{1,1}, y_{1,1})(x_{1,2}, y_{1,2}) \dots (x_{1,N}, y_{1,N})\}$ and $\mathcal{A}_{l,2} = \{(x_{2,1}, y_{2,1})(x_{2,2}, y_{2,2}), \dots (x_{2,N}, y_{2,N})\}$ be the first and second parents in the lth pair. The MINDIST normalization is carried out by rearranging the pairs in the second parent such that the sum distance between $\mathcal{A}_{l,1}$ and $\mathcal{A}_{l,2}$ is at minimum. The Euclidean sum distance has the following expression,

$$\sum_{n=1}^{N} \sqrt{(x_{2,n} - x_{1,n})^2 + (y_{2,n} - y_{1,n})^2}. \tag{2.13}$$

2.4.2.2 BLX-α Crossover

For the parent pair $\mathcal{A}_1 = \{J_1 J_2 ... J_{2N}\}$ and $\mathcal{A}_2 = \{L_1 L_2 ... L_{2N}\}$, the *blend crossover* (BLX) produces the offspring $\mathcal{A}' = \{M_1 M_2 ... M_{2N}\}$ by uniformly selecting the gene M_n from the range $[\min(J_n, L_n) - \alpha I, \max(J_n, L_n) + \alpha I]$, where $I = |J_n - L_n|$ and α is a design variable [3, 20]. A common value for α is 0.5.

2.4.2.3 Gaussian Mutation

The Gaussian mutation is performed on a gene J_i with a mutation rate P_m by

$$J_i' = \min(\max(\beta, J_{i,\min}), J_{i,\max}), \tag{2.14}$$

where $J_{i,\max}$ and $J_{i,\min}$ are the upper and lower bound on the gene J_i. Also, $\beta \sim N(\mu, \sigma)$ is a normally distributed random number with mean μ and standard deviation σ. In this work $\mu = 0$ and $\sigma = (J_{i,\max} - J_{i,\min})/10$.

2.4.3 Particle Swarm Optimization

Here we consider a simple form of PSO based on the work initially presented by Kennedy and Eberhart [21]. Each sensor in $\mathcal{S}$ represents a particle in the swarm.

Updating the velocity and position vectors for the nth particle is carried out as

$$V_n^{i+1} = wV_n^i + r_1c_1(P_n^{\text{best}} - P_n^i) + r_2c_2(P_G - P_n^i), \tag{2.15}$$

$$P_n^{i+1} = P_n^i + V_n^{i+1}, \tag{2.16}$$

where $i \in [0, i_{\max}]$ is the iteration counter and w is the inertia; its value changes with each iteration as $w = w_{\max} - (w_{\max} - w_{\min})\frac{i}{i_{\max}}$. The values of $w_{\min}$ and $w_{\max}$ are set to 0.4 and 0.9, respectively [22]. P_n^{best} is the point with the best fitness for the nth sensor, and P_G is the point with the best fitness among all sensors (particles). Velocity limits of $V_{\min} = -0.5(P_{\max} - P_{\min})$ and $V_{\max} = 0.5(P_{\max} - P_{\min})$ were used to initiate particles' velocities, where $P_{\min}$ and $P_{\max}$ are the ROI limits.

2.5 Simulation and Results

For our simulations, two different sets of parameters are used for the problem setup and the used algorithms—see Table 2.2. The first set is for generating results to compare the algorithms in the two-point path case, where no part of the terrain is conceived as an obstacle. The second set of parameters is mainly for investigating the A-star path case, where the presence of obstacles is possible. All terrains used throughout the simulations are synthetically generated. Several PCs were used to generate the simulations, with 8-16 GB RAM, CPU (i7-6700HQ 2.6 GHz, i7-4770 3.4

Table 2.2
Parameters Setup for the Two-Point and A-star scenarios.

		Two-Point	A-Star
Problem setup	$N_x \times N_y$	51×51	21×21
	N	$10 \rightarrow 70$	10
	R_s	5	3
	R_c	10	6
	r_e	$0.6R_s$	$0.6R_s$
	Pr_{th}	0.85	0.85
	a	0.5	0.5
	b	0.5	0.5
	α	0.9	0.9
	G_{th}	–	$[0.2, 0.4, 0.6, 0.8, > 1]$
AIS	P_s	50	15
	P_r	0.9	0.9
	P_c	0.1	0.1
	P_h	0.7	0.5
	P_m	0.01	0.05
	$g_{\max}$	100	30
PSO	P_s	50	15
	$w_{\min}$	0.4	0.4
	$w_{\max}$	0.9	0.9
	c_1	4	4
	c_2	2	2
	$Itr_{\max}$	100	30
NGA	P_s	50	15
	P_m	0.001	0.001
	$g_{\max}$	100	30

GHZ, i7-4770S 3.1 GHZ).

In the following we perform three sets of experiments to compare the presented algorithms. Table 2.3 outlines the scenarios for each of the experiments.

Table 2.3
Description of Performed Experiments

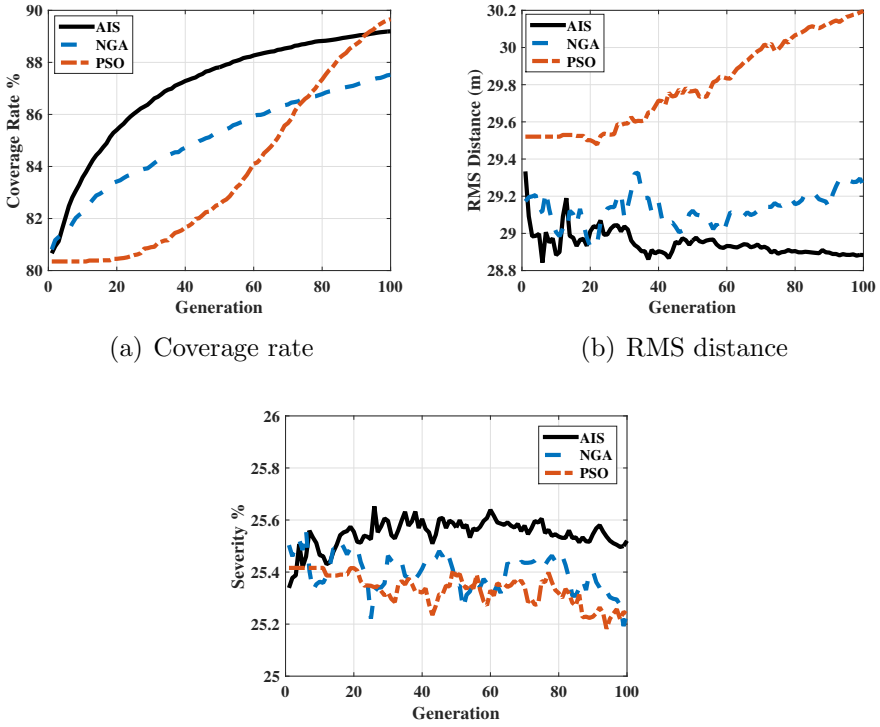
	Description	Parameters
Experiment 1	Obstacle-free ROI, where sensors follow a two-point path.	Table 1 (Two-Point)
Experiment 2	The ROI can contain obstacles, and sensors follow a path generated by the A-Star algorithm.	Table 1 (A-Star)
Experiment 3	Comparing the scenarios in experiments 1 and 2.	Table 1 (A-Star)

2.5.1 Experiment 1

In this experiment a comparison is held among the presented algorithms for the case where no part of the terrain is perceived as an obstacle; hence, sensors are able to follow a two-point path. The parameters used from Table 2.2 are the ones of the two-point case. The results shown in Table 2.4 are generated by averaging 50 runs of each algorithm, with $N = 70$. It can be seen from Table 2.4 that the AIS algorithm outperforms the other two in terms of traveled distance. Both the AIS and PSO algorithms seem to have better performance in terms of coverage rate than that of the NGA algorithm. With regards to the convergence rate, Fig. 2.3 reveals that for path severity the three algorithms are close to each other and decay at almost the same rate. The AIS algorithm has an obviously better performance in terms of the RMS distance, but slightly falls behind the PSO for the coverage rate especially after the 95th generation. It is worthwhile noticing that maximizing the coverage rate has its consequences over minimizing both the rms distance and the path severity, which

Table 2.4
Algorithms Comparison.

		Min.		RMS		Max.	
	$C_{\text{rate}}\%$	d	$Sev\%$	d	$Sev\%$	d	$Sev\%$
AIS	**89.2**	3.4	8.5	**28.9**	25.5	**54.4**	51
NGA	87.5	3.5	8.4	29.3	25.2	55.7	50.9
PSO	**89.7**	3.6	8.7	30.2	25.2	57.4	51.3



(a) Coverage rate

(b) RMS distance

(c) Severity

Figure 2.3: Convergence behavior of the optimization algorithms.

can be seen clearly from the divergence of the PSO for the rms distance, and the slow convergence of all three algorithms especially for path severity.

Figure 2.4 examines the impact of increasing the number of sensing nodes N on the coverage rate. The initial coverage rate is that of the first random deployment, and the optimal coverage rate is the one achieved by placing the sensing nodes with an

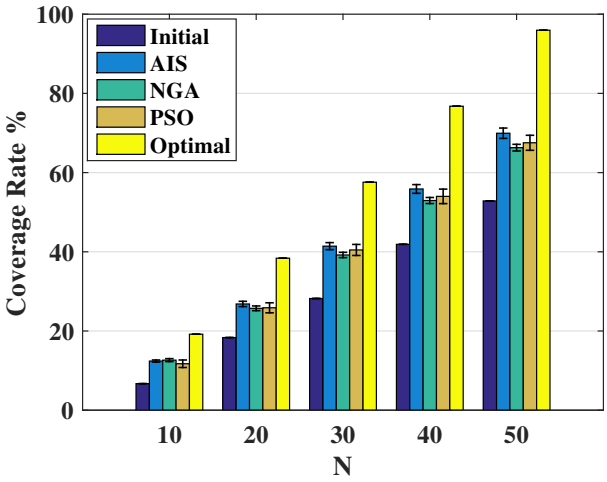


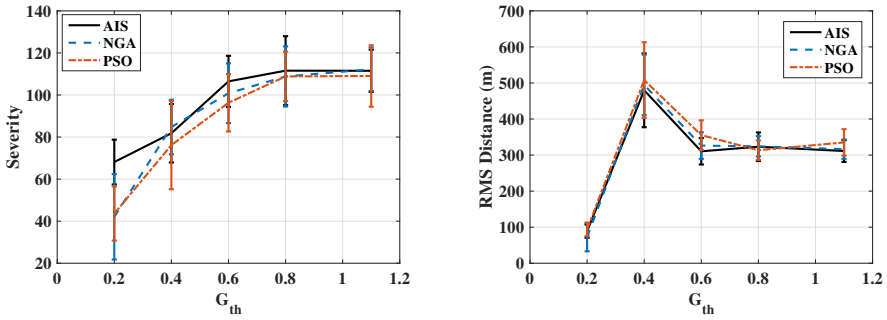
Figure 2.4: Coverage rate verses the number of sensing nodes N.

equal threshold distance d_{th} from each other within an obstacle-free ROI of a regular shape [2]. Furthermore, in the optimal case the nodes need to have similar sensing radius R_s, and $d_{\text{th}} = \sqrt{3}R_s$. All algorithms seem to achieve better coverage rate as N increases. Even though the PSO algorithm does not perform as well for $N < 30$, it soon recovers for higher number of sensing nodes. It is clear that the AIS algorithms achieves better coverage than the NGA and PSO algorithms, particularly for $N > 10$. From the previous, the AIS algorithms seems to offer the best compromise; good coverage rate, low rms traveling distance and a comparable path severity.

2.5.2 Experiment 2

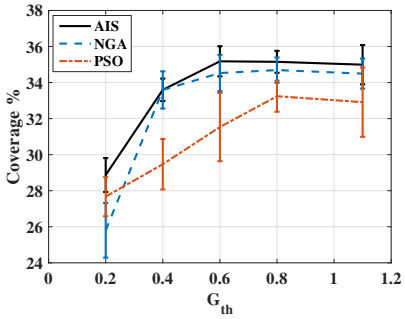
Here we address the scenario where parts of the terrain can be considered as obstacles. The A-star algorithm is used to generate a relocation path capable of navigating

around obstacles. At any generation/iteration, if the solution produces a destination point that is surrounded by obstacles, over an obstacle or on the initial location, no relocation occurs. The parameters used to generate the results for this experiment are found under the "A-star" column in Table 2.2. Figure 2.5 illustrates the impact of varying G_{th} on the total severity, total RMS distance, and coverage rate. The total severity is evaluated by summing Sev_{RMS} over all generations and the total distance is evaluated likewise. As observed in Fig.2.5, the three algorithms acquire similar performances in terms of severity and RMS distance. Both the AIS and NGA algorithms outperform the PSO algorithm in terms of coverage rate, especially for $G_{th} > 0.2$, Fig.2.5(c). The closeness in performance between the AIS and NGA can be referred to the somewhat similar evolutionary framework, as opposed to the swarm evolutionary approach the PSO algorithm is based upon. Moreover, even though the error bars of the PSO coverage curve seem substantial, performing a z-test confirms that it belongs to different distribution . As G_{th} increases the number of obstacles decrease, which in effect gives more freedom for the nodes to move around, generating a longer relocation paths. A longer path means larger RMS distance and higher probability of traversing more severe terrains. It is observed from Fig.2.5(b) that the RMS distance peak at $G_{th} = 0.4$, this is due to obstacles that force the nodes to travel longer relocation paths, see Fig.2.8(b).



(a) Total severity

(b) Total RMS distance

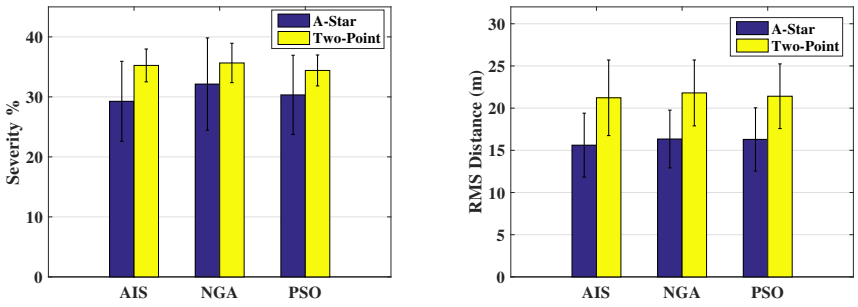


(c) Coverage rate

Figure 2.5: Impact of the gradient threshold G_{th}.

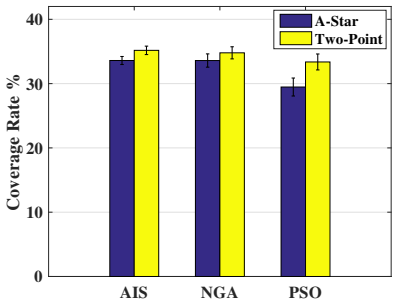
2.5.3 Experiment 3

For this experiment, a comparison is made between the two scenarios presented in the first two experiments, but with the parameters of the second experiment applied to both. In Fig. 2.6 a comparison between the utilized algorithms is held for total severity, total RMS distance, and coverage rate with $G_{\text{th}} = 0.4$ for the A-star scenario. Due to the presence of obstacles in the A-Star scenario, the movement of nodes is limited and hence shorter relocation paths are generated, Fig.2.6(b). A shorter path means less terrain to be traversed, resulting in a lower path severity, Fig.2.6(a). The



(a) Severity

(b) RMS distance



(c) Coverage rate

Figure 2.6: Comparison of the two-point and A-Star scenarios. For the A-star scenario the gradient threshold was $G_{\text{th}} = 0.4$.

lower path severity and RMS distance for the A-Star scenario comes at the expense of reduced coverage rate, Fig. 2.6(c). For both scenarios the AIS algorithm seems to outperform the other two in every aspect under consideration.

The execution time of the three algorithms is compared for $G_{\text{th}} > 1$ reflecting the case of an obstacle free ROI, Fig. 2.7. .From the figure, the AIS algorithm possesses the highest execution time while the PSO has the lowest. Taking the two-point scenario into consideration and with same parameters as in experiment 2, all three algorithms execute in a time under 0.9 seconds—for $G_{\text{th}} > 1$—which is around 95% less. For

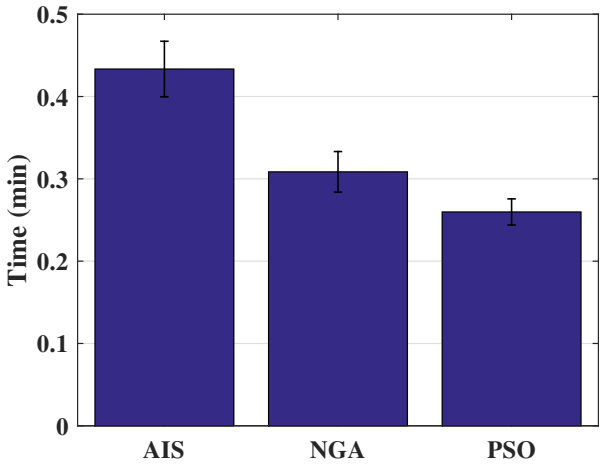
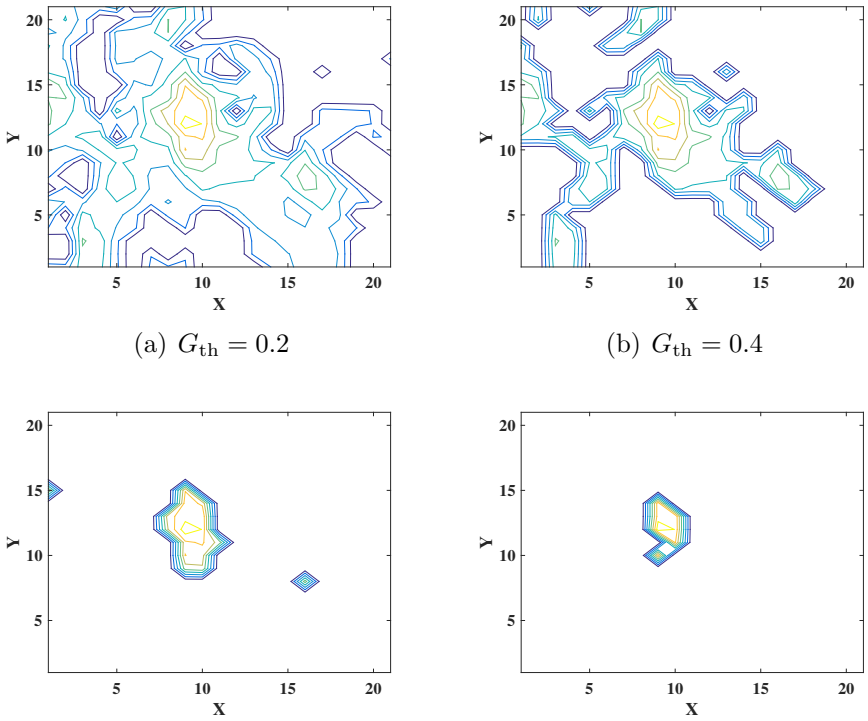


Figure 2.7: Comparing the execution time for the three algorithms. Here the gradient threshold was $G_{\text{th}} > 1$, meaning an obstacle free ROI.

no obstacles present in the ROI, and in the case where the execution time is not of essence, the AIS algorithm would be more preferable, for it has a good coverage rate with reasonable RMS distance and severity, especially in the two-point case.

2.6 Conclusion

For a WSN where the sensor nodes are initially deployed randomly in a ROI of a regular shape, this work addressed the problem of sensor relocation to maximizing the sensing coverage, while maintaining a minimal mobility cost. The cost of mobility was directly associated to the traveling distance and the severity of the relocation path. The terrain severity was characterized based on its gradient. Two main scenarios were investigated: the first assumed that the sensor nodes are capable of traversing



(a) $G_{\text{th}} = 0.2$ (b) $G_{\text{th}} = 0.4$

(c) $G_{\text{th}} = 0.6$ (d) $G_{\text{th}} = 0.8$

Figure 2.8: Obstacles in the ROI for different gradient threshold G_{th} values. As G_{th} increases, the number of obstacles in the ROI decreases.

any terrain along their path, and the second considered the case where nodes might not be able to pass through certain terrains, which gives rise to obstacles. In the first scenario the relocation path was considered as a straight line defined by two points, and in the second the A-star algorithm was used to generate a path capable of maneuvering obstacles. Results show that there is no single algorithm that surpasses the others in all cases and scenarios. Having said that, the AIS algorithm seems to offer the best compromise between coverage rate and cost of mobility. Even though the AIS algorithm has the longest execution time, for the relocation problem this might not be a crucial factor. Table 2.5 provides the advantages and disadvantages

Table 2.5
Advantages and Disadvantages of the Presented Algorithms

	Advantages	Disadvantages
AIS	Best coverage rate for most values of N. Low traveling distance.	Slightly higher path severity than NGA and PSO. High execution time.
NGA	Good coverage rate especially for low N.	High execution time, but lower than AIS.
PSO	Low execution time.	In general has lower coverage rate.

of the presented algorithms in the context of our problem.

Chapter 3

Sensor Relocation for Improved Target Tracking

3.1 Introduction

Due to its importance and influence on the dependability of a *wireless sensor network* (WSN), the coverage problem has attracted much attention in the literature. Depending on the application, sensor deployment can be oriented toward improving *area* coverage [2, 23, 24, 25, 26, 27], or *target/event* coverage [18, 28, 29, 30]. In harsh or hostile environments a controlled deployment can be arduous and in some scenarios

not possible. In such a case, the only alternative would be a WSN with randomly distributed sensing nodes, resulting in coverage holes and connectivity problems among the nodes [31]. Hence, a relocation of the sensing nodes might be necessary after the initial deployment to improve the performance of the network. In this note, target tracking is of interest and the quality of tracking is a main objective. Targets usually tend to follow patterns while traveling in a given region; animals travel in routes for food, mating or shelter; troops and artillery in a war zone travel in secure but traverse-able paths. Therefore, it is of interest to learn target trends and relocate the appropriate sensor nodes accordingly. Furthermore, it might be desirable to enhance the tracking quality in a given *region of interest* (ROI) due to the region's strategic importance. However, sensor relocation to cover a certain ROI creates coverage holes in other parts of the field, preventing the detection of possible future targets. Hence, it is also important to keep the field coverage rate into consideration. The mobility of sensors is usually expensive in terms of power consumption, therefore it is desirable to keep this cost to a minimum while relocating the nodes. The problem at hand can be related to set-coverage problems which are considered NP-complete [14]. As a result, the use of an evolutionary computation algorithm would be reasonable. For this work, the *genetic algorithm* (GA) is considered.

The coverage and monitoring of events and targets have received considerable attention in the literature. A decentralized event-based self deployment algorithm was presented in [28]. The authors used a virtual force algorithm for relocating sensing

nodes around the event location with a sensor density that depends on the event intensity. The coverage and monitoring of a large population of objects—*mass objects*—was investigated from a probabilistic perspective [30]. The authors introduced an online distributed recursive *expectation maximization* (EM) algorithm for dynamic deployment of sensors with the purpose of improving the detection and boundary estimation of mass objects.

Olfati introduced the flocking concept to the distributed tracking of a target [32]. Even though the flocking behaviour of the mobile sensor network was a byproduct of minimizing the estimation error for each of the nodes, it resulted in a connected network with topologies that improved the performance of the *distributed Kalman filter* (DKF).

Learning contextual information from an image sequence for improving object-tracking has also been addressed [33, 34]. *Gaussian mixture models* (GMMs) were used to learn the distributions of the object-birth and clutter events [34]. The learned models were used to adapt the object-tracking filter for an improved performance.

In this work, for a WSN with an initial deployment following a uniform distribution, a *distributed extended information filter* (DEIF) is used to perform target tracking. The EIF is a variant of the Kalman filter that has been used in the literature in its distributed form for target tracking [35, 36, 37]. Also, the *global node selection* (GNS) algorithm is used at each dynamic step to select a set of active sensors for target

localization at the next dynamic step [35]. For the problem at hand, the *geometric dilution of precision* (GDOP) is used as a basis in the sensor selection algorithm and as an one of the objective functions considered for the sensor relocation process. The GDOP is a dimensionless measure originally used in *global positioning system* (GPS) to select satellites with geometrical positions that offer a more accurate localization [38]. Most related work in the literature only considers optimizing the deployment of the WSN for achieving a better tracking [39, 40, 41]. This has the problem of not being able to adapt to the changes in targets' characteristics and mobility trends. This work addresses this issue by introducing a relocation algorithm that moves the sensor nodes to a ROI that corresponds to the mobility patterns of the target. Moreover, the impact of the terrain was reflected on the generated ROI. In general, the contribution of this work is summarized by,

† Presenting a relocation algorithm for improving the target tracking accuracy.

† Relocating the sensor nodes to a region of interest that was deduced based on targets mobility trends.

† The presented algorithm is dynamic in the sense that it is capable of relocating the nodes as the targets' mobility trends changes.

† The use and comparison of multiple relocation criteria that showed improvement in the root mean square positioning error.

Table 3.1
Notations Used

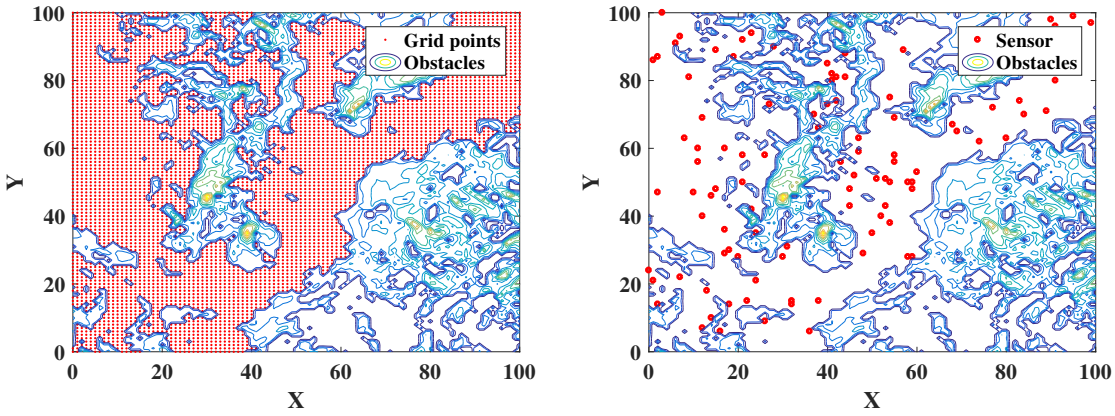
Notation	Definition	Notation	Definition
$\mathbf{s}$	State vector	(P_x, P_y)	Target coordinates
(v_x, v_y)	Velocity vector	$\mathbf{u}$	Excitation noise
$\mathbf{z}$	Measurements vector	$\mathbf{h}(.)$	Nonlinear measurement model
$\{R, \beta\}$	Range and bearing measurements	$\boldsymbol{\varphi}$	Measurement noise
$\mathbf{C}$	Measurement noise covariance	$\boldsymbol{\Omega}$	Information matrix
$\boldsymbol{\eta}$	Information vector	$\boldsymbol{P}$	Covariance matrix
$\mathbf{H}$	Gradient of $\boldsymbol{h}(.)$	$\{\boldsymbol{J}_f, \boldsymbol{J}_p, \boldsymbol{J}\}$	Posterior, prior and measurement FIMs
$\mathcal{S}_a$	Set of active sensors	$\rho(\mathcal{S}_a)$	MS position error
$\mathcal{G}$	Set of all grid points	R_{cov}	Coverage rate
p_c	Probability of crossover	p_m	Probability of mutation
N_{pop}	Population size	$g_{\max}$	Max. number of generations

Table 3.1 lists the main notations used throughout this work. The remainder of this chapter is organized as follows. Section 3.2 describes the problem structure as well as the adopted assumptions. The tracking algorithm is presented in Section 3.3. Section 3.4 discusses the formation of the ROI using the *kernel density estimator* (KDE) method. The process of sensor relocation and the objective functions used for location optimization are presented in Section 3.5. Section 3.6 describes the experiments performed and offers a discussion for the presented results. Finally, the work is concluded in Section 3.7.

3.2 Problem Statement and Assumptions

In an $N_x \times N_y$ field, a WSN of N sensors (nodes) is deployed where the locations of the nodes first follow a uniform distribution. Sensors are assumed to be mainly distributed in the largest traverse-able region in the field, avoiding obstacles and locked regions, e.g., see Fig. 3.1. This is a fairly realistic assumption for large fields, as sensors tend to be deployed in regions that are predominantly traverse-able by targets as well as mobile sensor platforms. The set $\mathcal{S}$ includes all sensors in the WSN. Sensors are assumed to be time synchronised and each sensor knows the location of all other nodes. The connectivity among the nodes is assumed to be maintained through a multi-hop routing model [42], the details of which are beyond the scope of this chapter. For the purpose of investigating the problem of node relocation, a simple model of a single target with a nearly constant velocity is adopted. Moreover, the measurement model assumes a stationary target. A probabilistic detection model is used, with a probability of one within a predetermined range from the sensor, that afterwards drops exponentially until it reaches zero. The flow diagram for the problem structure is illustrated in Fig. 3.2. Initially, the randomly distributed WSN performs tracking over a given period of time that depends on the activity and nature of targets moving in the field. The purpose of this phase is to collect a database of location estimates for all the tracked targets—see Section 3.3. The gathered data are used in the second phase to fit a model using the KDE algorithm. This model

is used to specify a ROI of high probability target locations; more details are found in Section 3.4. Finally, the relocation algorithm moves the sensor nodes in a manner that would improve the tracking performance in the constructed ROI; see Section 3.5. Appendix A provides the derivation of the mean square position error, while Appendix B presents a brief discussion of the error propagation throughout the proposed system. The time complexity analysis can be found in Appendix C.



(a) Largest traverse-able region (b) Sensors initial deployment

Figure 3.1: Initial deployment of sensor nodes in the field's largest traverse-able region.

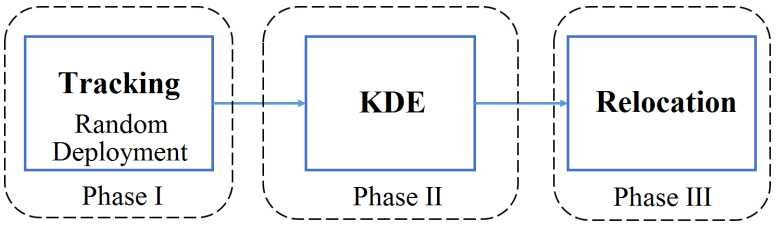


Figure 3.2: System flow diagram.

3.3 Tracking Algorithm

For collective tracking of a given target, two main procedures are essential for the tracking algorithm. The first selects a set of sensor nodes used for estimating the target state, while the other carries out the estimation. Next is a brief description of the DEIF algorithm used for target state estimation, and the GNS algorithm used for sensor node selection.

3.3.1 Distributed Extended Information Filter

In this section, a set of equations used for the estimation of a target location using a set of active senors $\mathcal{S}_a$ is presented. The target non-linear dynamic system model, composed of a state equation and measurement equation, is now described. The *state equation* is

$$\mathbf{s}_k = \mathbf{G}\mathbf{s}_{k-1} + \mathbf{B}\mathbf{u}_k, \tag{3.1}$$

where the state vector $\mathbf{s}$ is

$$\mathbf{s} = \begin{bmatrix} P_x \\ P_y \\ v_x \\ v_y \end{bmatrix},$$

(P_x, P_y) are the target coordinates and (v_x, v_y) are the velocity vector components. Also, $\mathbf{u} \sim N(\mathbf{0}, \sigma_{\mathrm{u}}^2 \mathbf{I})$ is a representation of the target acceleration. Matrices $\mathbf{G}$ and $\mathbf{B}$ are

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0.5\Delta^2 & 0 \\ 0 & 0.5\Delta^2 \\ \Delta & 0 \\ 0 & \Delta \end{bmatrix},$$

where Δ is the step size between two time snapshots. The *measurement equation* is

$$\mathbf{z}_k^{(j)} = \mathbf{h}^{(j)}(\mathbf{s}_k) + \boldsymbol{\varphi}_k, \tag{3.2}$$

where the (j) superscript indicates the jth sensor in the set $\mathcal{S}_a$.The function $\mathbf{h}(.)$ is

$$\mathbf{h}(\mathbf{s}) = \begin{bmatrix} \sqrt{(P_x - x)^2 + (P_y - y)^2} \\ \arctan \frac{P_y - y}{P_x - x} \end{bmatrix} = \begin{bmatrix} R \\ \beta \end{bmatrix},$$

which represents the nonlinear measurement model, such that R and β are the range and bearing from a sensor at (x, y) to the target at an estimated position of (P_x, P_y), respectively. Also, $\boldsymbol{\varphi}_k \sim N(\mathbf{0}, \mathbf{C})$ is the measurements noise, where

$$\mathbf{C} = \begin{bmatrix} \sigma_R^2 & 0 \\ 0 & \sigma_\beta^2 \end{bmatrix}.$$

Here, σ_R^2 and σ_β^2 are fixed among the sensors at all times. Even though the measurement model $\mathbf{h}(.)$ does not take into consideration the target mobility, the simulated bearing measurements did incorporate the target non-stationarity, see equation (3) in [35]. For the case where range measurements are acquired via a Lidar or a Radar, the impact of mobility is minimal, hence the simulated range measurements assumed a stationary target.

Next, a set of equations for the prediction and correction stages are presented. In the first stage a prediction for the covariance matrix $\mathbf{P}$ and the state vector $\boldsymbol{s}$ is evaluated. The second stage provides the estimation by correcting the predictions using the current measurement. The *prediction* is

$$\bar{\mathbf{s}}_k = \mathbf{G}\hat{\mathbf{s}}_{k-1}, \tag{3.3}$$

$$\bar{\mathbf{P}}_k = \mathbf{G}\mathbf{P}_{k-1}\mathbf{G}^T + \sigma_u^2\mathbf{B}\mathbf{B}^T. \tag{3.4}$$

And the *correction* is

$$\hat{\mathbf{s}}_k = \mathbf{P}_k\left(\bar{\mathbf{P}}_k^{-1}\bar{\mathbf{s}}_k + \sum_{j\in\mathcal{S}_a}\boldsymbol{\eta}_k^{(j)}\right), \tag{3.5}$$

$$\mathbf{P}_k^{-1} = \bar{\mathbf{P}}_k^{-1} + \sum_{j\in\mathcal{S}_a}\boldsymbol{\Omega}_k^{(j)}, \tag{3.6}$$

where $\boldsymbol{\eta}$ and $\boldsymbol{\Omega}$ are the information vector and information matrix, respectively. These

are expressed as

$$\boldsymbol{\eta}_k^{(j)} = \mathbf{H}^{(j)T}\mathbf{C}^{-1}\left(\mathbf{e}_k^{(j)} + \mathbf{H}^{(j)}\bar{\mathbf{s}}_k\right), \tag{3.7}$$

$$\boldsymbol{\Omega}_k^{(j)} = \mathbf{H}^{(j)T}\mathbf{C}^{-1}\mathbf{H}^{(j)}, \tag{3.8}$$

such that $\mathbf{e}_k^{(j)} = \mathbf{z}_k^{(j)} - \mathbf{h}(\bar{\mathbf{s}}_k^{(j)})$ is the measurement residual. $\mathbf{H}$ is the gradient of the nonlinear measurement model $\mathbf{h}(.)$ at $\bar{\mathbf{s}}_k$, and it can be easily verified that it has the following expression,

$$\mathbf{H} = \begin{bmatrix} \frac{P_x - x}{R} & \frac{P_y - y}{R} & 0 & 0 \\ \frac{-(P_y - y)}{R^2} & \frac{P_x - x}{R^2} & 0 & 0 \end{bmatrix} = \begin{bmatrix} \cos\beta & \sin\beta & 0 & 0 \\ \frac{-\sin\beta}{R} & \frac{\cos\beta}{R} & 0 & 0 \end{bmatrix}.$$

Since the DEIF algorithm is recursive, the complexity of a single run represents the algorithm complexity. Following a fairly simple analysis of the algorithm, a complexity of $O(n_s^3 + Mn_s^2 + n_m n_s^2)$ is achieved, where $n_s = |\mathbf{s}|$, $n_m = |\mathbf{h}(\mathbf{s})|$ and $M = |\mathcal{S}_a|$. In our work $n_m < n_s$, hence the complexity further simplifies to $O(n_s^3 + Mn_s^2)$. Refer to Appendix C for details.

3.3.2 Active Set Selection

We now present the GNS algorithm, modified to use both the bearing and range measurements to provide $\mathcal{S}_a$. At each time step the GNS algorithm selects the set of nodes $\mathcal{S}_a$ which is used to localize the target at the upcoming time-step. Also, the algorithm considers only sensors within a sensor's detection range from the last estimate of the target location. This range is referred to as R_{GNS}. From one aspect this leads to lower computational complexity, but from another it can result in an empty set for $\mathcal{S}_a$, forcing missing estimates for parts of the target trajectory. It will be shown that target relocation generally minimizes the occurrence of this problem. The objective of the GNS algorithm is to select a set that will minimise the expected *mean-squared* (MS) position error of the target. In the following we derive an expression for the MS position error. Equation (3.6) can be rewritten as

$$\mathbf{J}_f = \mathbf{J}_p + \sum_{j \in \mathcal{S}_a} \mathbf{H}^{T^{(j)}} \mathbf{C}^{-1^{(j)}} \mathbf{H}^{(j)}, \tag{3.9}$$

$$\mathbf{H}^{T^{(j)}} \mathbf{C}^{-1^{(j)}} \mathbf{H}^{(j)} = \begin{bmatrix} \frac{\cos^2 \beta^{(j)}}{\sigma_R^2} + \frac{\sin^2 \beta^{(j)}}{R^{2^{(j)}} \sigma_\beta^2} & \frac{\cos \beta^{(j)} \sin \beta^{(j)}}{\sigma_R^2} - \frac{\sin \beta^{(j)} \cos \beta^{(j)}}{R^{2^{(j)}} \sigma_\beta^2} & 0 & 0 \\ \frac{\cos \beta^{(j)} \sin \beta^{(j)}}{\sigma_R^2} - \frac{\sin \beta^{(j)} \cos \beta^{(j)}}{R^{2^{(j)}} \sigma_\beta^2} & \frac{\sin^2 \beta^{(j)}}{\sigma_R^2} + \frac{\cos^2 \beta^{(j)}}{R^{2^{(j)}} \sigma_\beta^2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.10}$$

where $\mathbf{J}_f = P_k^{-1}$ is the posterior (filtered) *Fisher information matrix* (FIM), and $\mathbf{J}_p = \bar{P}_k^{-1}$ is the prior FIM. The $[\mathbf{H}^{T^{(j)}}\mathbf{C}^{-1^{(j)}}\mathbf{H}^{(j)}]$ expression in (3.9) can be easily expanded to the form presented in (3.10). This results in

$$\mathbf{J}_f = \mathbf{J}_p + \begin{bmatrix} \sum_{j\in\mathcal{S}_a} \mathbf{J}^{(j)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \Leftrightarrow \mathbf{J}_f = \mathbf{J}_p + \begin{bmatrix} \mathbf{J} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{3.11}$$

where $\mathbf{J}$ represent the measurements FIM, and its expression is a result of the both, the form of the observation/measurement model $\mathbf{h}(\boldsymbol{s})$, and the process of evaluating the measurement FIM as seen from equation (3.10). Now, the expected MS position error can be expressed as,

$$\rho\left(\mathcal{S}_a\right) = [\mathbf{J}_f^{-1}]_{1,1} + [\mathbf{J}_f^{-1}]_{2,2}, \tag{3.12}$$

which can be further simplified to [35]

$$\rho(\mathcal{S}_a) = \frac{\mathrm{tr}\{\mathbf{J}\} + \mathrm{tr}\{\tilde{\mathbf{J}}_p\}}{\det\{\mathbf{J}\} + \det\{\tilde{\mathbf{J}}_p\} + \Lambda}, \tag{3.13}$$

where $\tilde{\mathbf{J}}_p = [\mathbf{J}_p]_{1:2,1:2} - [\mathbf{J}_p]_{1:2,3:4}[\mathbf{J}_p]_{3:4,3:4}^{-1}[\mathbf{J}_p]_{1:2,3:4}^T$, and $\Lambda = [\mathbf{J}]_{1,1}[\tilde{\mathbf{J}}_p]_{2,2} + [\mathbf{J}]_{2,2}[\tilde{\mathbf{J}}_p]_{1,1} - 2[\mathbf{J}]_{1,2}[\tilde{\mathbf{J}}_p]_{2,1}$. The details to reach the expression at (3.13) can be found in Appendix

A. In case the prior FIM $\mathbf{J}_p$ was ignored, the expression in (3.13) will reduce to

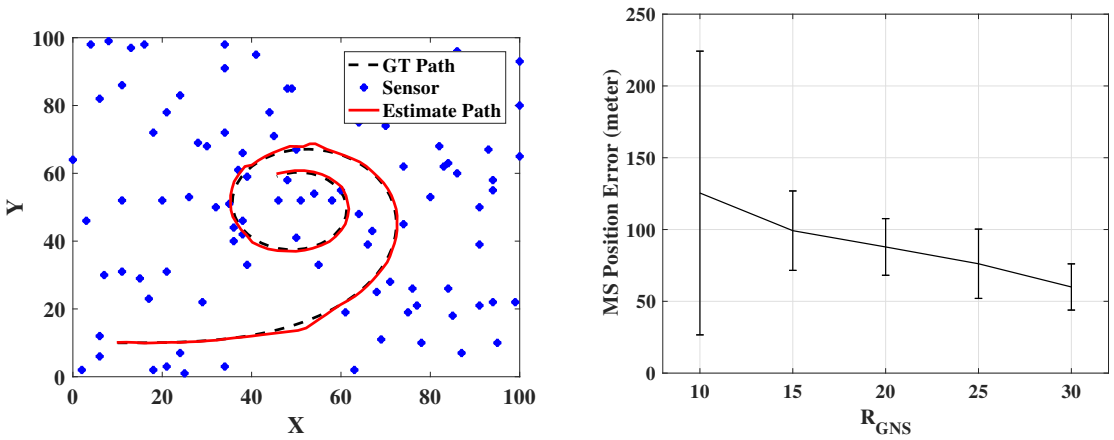
$$\rho(\mathcal{S}_a) = \frac{\text{tr}\{\mathbf{J}\}}{\det\{\mathbf{J}\}}. \tag{3.14}$$

This form of the MS position error reflects the GDOP measure [43]. In relocating the sensors we will be looking at minimizing the GDOP measure as presented in [44] with an effort to reduce the MS position error; see Section 3.5.

The GNS algorithm has two phases, the first is referred to as *Add one sensor at a time*, and the second is the *Simplex*. In the first phase, a greedy algorithm that minimizes the MS position error adds one sensor at a time to the active set. Prior to applying the greedy algorithm, the active set is initialized by performing an exhaustive search to find the two sensors that minimize (3.13). The greedy algorithm performs the following [35],

$$j = \arg \min_{j \in \mathcal{S} \setminus \mathcal{S}_a} \rho(\{j \cup \mathcal{S}_a\}). \tag{3.15}$$

Equation (3.15) is performed iteratively until $|\mathcal{S}_a| = M$. The *Simplex* algorithm goes over each sensor in the active set evaluated in the first phase and tries to find a replacement inactive sensor that would result in a smaller $\rho(\mathcal{S}_a)$. Next, the impact of the range R_{GNS} on the MS position error is explored for a simple case of tracking a target on a spiral path; see Fig. 3.3. It is clear that increasing the range decreases



(a) Tracking on a spiral path. $R_{\mathrm{GNS}} = 12$

(b) MS position error Vs. R_{GNS}

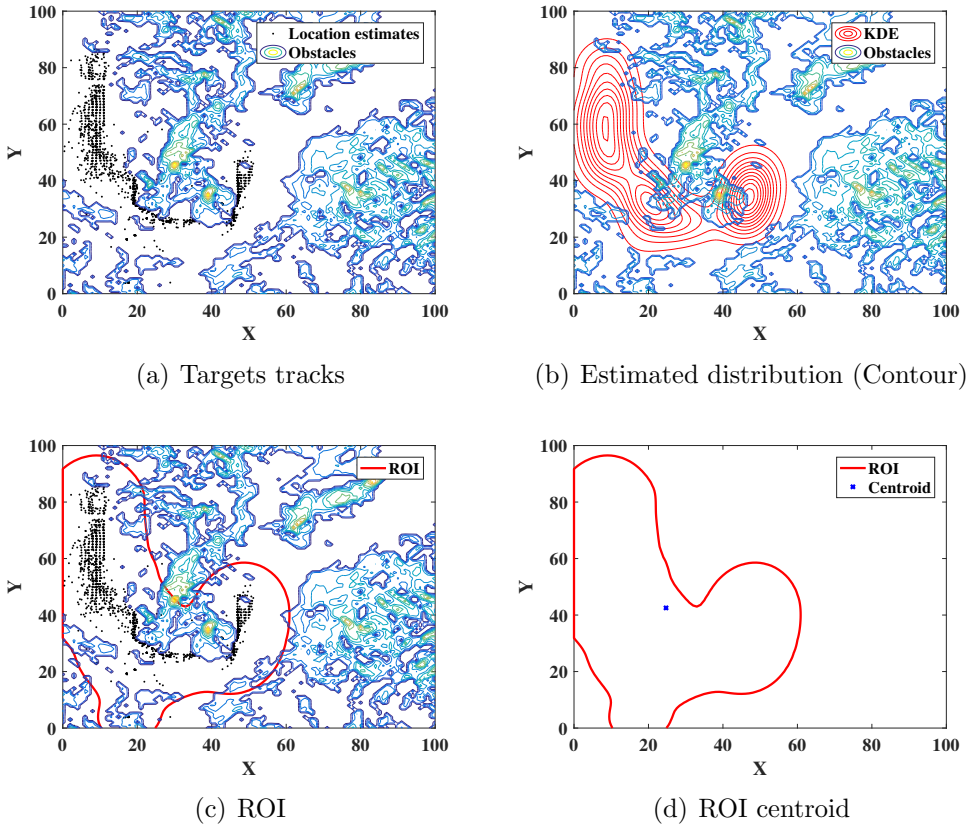
Figure 3.3: The impact of R_{GNS} on the MS position error.

the MS position error. Since the original GNS algorithm tends to select the nodes that are close to the last target estimate, our modified GNS algorithm performance reaches that of the original at $R_{\mathrm{GNS}} > 12$.

3.4 ROI Formation

Before starting the relocation process, the ROI needs to be established. In this work, the ROI is formed based on an estimated probability density model for the tracked targets' positions. Initially, the randomly distributed WSN tracks the targets traversing the field, providing a database of target location estimates. In this work targets are tracked individually. For the case where a target enters a coverage hole, the location estimate is considered missing for that time step; see Fig. 3.4(a). Even

with the effect of coverage holes, the database reflects the trends of tracks preferable by the targets. The collected training data is fed to the KDE algorithm resulting in a probability density function estimate, the boundaries of which form the ROI; see Fig. 3.4. Also, see phases I and II in Fig. 3.2. The KDE algorithm can be either performed in a centralized fashion at a node equipped with higher computational resources, or in a decentralized manner [45].



(a) Targets tracks

(b) Estimated distribution (Contour)

(c) ROI

(d) ROI centroid

Figure 3.4: The formation of the ROI based on a fitted model of the estimated targets' locations.

3.4.1 Kernel Density Estimation

The KDE is a non-parametric method used for the estimation of an unknown probability density function. For the problem at hand, the estimation of a density function based on the location estimates is of interest. Let $\left\{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_n \leftrightarrow \mathbf{X}_i = [P_x^{(i)}, P_y^{(i)}]\right\}$ represent the estimated target locations from phase I, where they are assumed to be independent and identically distributed. The density estimator is expressed as

$$p(\mathbf{X}, \nu) = \frac{1}{n\nu} \sum_{i=1}^{n} K\left(\frac{\mathbf{X} - \mathbf{X}_i}{\nu}\right), \tag{3.16}$$

where ν is a smoothing parameter, also referred as the bandwidth. $K(.)$ is a kernel function that i) is symmetric around zero, ii) integrates to one, and iii) is non-negative.

For simplicity, the Gaussian radial basis function is chosen as the kernel. Also, the bandwidth ν is selected based on the rule-of-thumb approach [46],

$$\nu = \left(\frac{4\sigma_{\mathbf{X}}^5}{3n}\right)^{1/5} \approx 1.06\sigma_{\mathbf{X}} n^{-1/5}, \tag{3.17}$$

where $\sigma_{\mathbf{X}}$ is the standard deviation of the locations data. Now that we have a way of learning a probabilistic model for target locations, we can dynamically relocate the sensors to maximize their ability to detect and track these targets.

3.5 Senors Relocation

3.5.1 Sensors Attraction

Sensor attraction refers to the process of increasing the density of sensors in the ROI. Sensors that are outside the ROI and nearest to its centroid are considered for attraction; see Fig. 3.4(d). Assuming the ROI to be always a non-intersecting polygon with m vertices, the centroid of the ROI is computed as [47],

$$C_x = \frac{1}{6A_{\text{ROI}}} \sum_{i=0}^{m-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$
$$C_y = \frac{1}{6A_{\text{ROI}}} \sum_{i=0}^{m-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

such that,

$$A_{\text{ROI}} = 0.5 \sum_{i=0}^{m-1} (x_i y_{i+1} - x_{i+1} y_i)$$

is the area of the ROI.

3.5.2 Sensor Position Optimization

The next step is to optimize the location of the sensors within the ROI to minimize an objective function. Mainly, two fitness functions are considered in this work, GDOP and K-coverage. The GDOP measures the goodness of the sensors' geolocation around a target to achieve accurate localization, and K-coverage makes sure that each coordinate point is covered by at least K sensors. Hence, a set of *virtual target points* (VTPs) need to be placed inside the ROI around which the optimization is performed. For this effort, a grid of equally spaced points is laid inside the ROI. The density of this grid is a design parameter. The number of nodes attracted to the ROI depends on the following ratio,

$$R_{\text{attraction}} = \frac{|\mathcal{S}_{\text{final}}|}{N_{\text{VTP}}}, \tag{3.18}$$

where $|\mathcal{S}_{\text{final}}|$ is the final count of sensors in the ROI after attraction and N_{VTP} is the number of VTPs in the ROI. Therefore, for a given attraction ratio the number of sensors to be attracted is

$$N_{\text{attract}} = |\mathcal{S}_{\text{final}}| - |\mathcal{S}_{\text{initial}}|,$$

where $|\mathcal{S}_{\text{initial}}|$ is the number of sensors inside the ROI before attraction.

3.5.3 Fitness Functions

In this section, three fitness functions are presented. The first is the GDOP measure used to optimize the location of the sensors inside the ROIs. The second is the coverage rate, and is used to relocate sensors outside the ROIs to mend/reduce the coverage holes produced from both the initial random deployment and the sensor attraction procedure. The third is the mobility cost, represented by the *root mean square* (RMS) distance of the mobilized nodes. In this work, a GA is used to optimize the objective function.

3.5.3.1 geometric Dilution of Precision

As a dimensionless quantity, the GDOP represents the relationship between the range measurement error and the target location error. The smaller the GDOP value, the better the positioning. For 2D scenarios, the GDOP can be expressed as [44]

$$\text{GDOP} = \sqrt{[Q]_{1,1} + [Q]_{2,2}}, \tag{3.19}$$

such that

$$Q = (A^T A)^{-1},$$

where A is the Jacobian matrix of the noise-free range measurements,

$$A = \begin{bmatrix} \frac{P_x - x_1}{R_1} & \frac{P_y - y_1}{R_1} \\ \frac{P_x - x_2}{R_2} & \frac{P_y - y_2}{R_2} \\ & . \\ & . \\ \frac{P_x - x_M}{R_M} & \frac{P_y - y_M}{R_M} \end{bmatrix},$$

and R_m is the range from the m'th sensor to target. $M = |\mathcal{S}_a|$ is the number of sensors in the active set. The GDOP fitness function is represented as the average of all the GDOP values evaluated around the VTPs in the ROI,

$$F_{\text{GDOP}} = \frac{1}{N_{\text{VTP}}} \sum_{i=1}^{N_{\text{VTP}}} \text{GDOP}_i \tag{3.20}$$

where GDOP_i is the GDOP value around the ith VTP.

3.5.3.2 Coverage Rate

As previously mentioned, coverage holes outside the ROI need to be addressed. This can be done by relocating the senors that were not involved in the processes of attraction and location optimization inside the ROI. The objective of this relocation is to maximize the coverage rate. For this we follow a probabilistic sensing model,

where each sensor has a circular sensing/detection area with a radius of R_s [2],

$$p^{(i)}(X) = \begin{cases} 1 & R_s - r_e \geq d\{(x_i, y_i), X\}, \\ \exp^{-\alpha\gamma^{\xi}} & R_s - r_e < d\{(x_i, y_i), X\} < R_s + r_e, \\ 0 & R_s + r_e \leq d\{(x_i, y_i), X\}, \end{cases} \tag{3.21}$$

where $p^{(i)}(X)$ is the probability that the grid point X is covered by the ith sensor. Also, r_e is the uncertainty (error) in the sensing range R_s. $d\{(x_i, y_i), X\}$ represents the distance between the ith sensor and the grid point X. In the uncertainty region—i.e., $R_s - r_e < d\{(x_i, y_i), X\} < R_s + r_e$—the coverage probability is exponentially decaying, where $\gamma = d\{(x_i, y_i), X\} - (R_s - r_e)$ and $\{\alpha, \xi\}$ are control parameters. Now the coverage rate of the whole field can be expressed as

$$R_{\text{cov}} = \frac{A_{\text{cov}}}{A_{\text{total}}}, \tag{3.22}$$

where the total area of the field is $A_{\text{total}} = N_x \times N_y$, and the area covered by the sensors is

$$A_{\text{cov}} = \sum_{X \in \mathcal{G}} p^{(\mathcal{S})}(X), \quad \forall p^{(\mathcal{S})}(X) \geq p_{\text{th}},$$

such that $\mathcal{G}$ is the set of all grid points in the field. $p^{(\mathcal{S})}(X) = 1 - \prod_{i=1}^{|\mathcal{S}|}(1 - p^{(i)}(X))$ is the coverage of grid point X considering all sensors in $\mathcal{S}$. p_{th} is a probability

threshold that governs the impact of $p^{(\mathcal{S})}(X)$ on the computation of A_{cov}, which is a design parameter. Hence, the fitness would be the rate of uncovered area,

$$F_{\text{uncov}} = 1 - R_{\text{cov}}. \tag{3.23}$$

3.5.3.3 Mobility Cost

The mobility cost is utilized to penalize the movement of the sensors, with an effort to minimize the average distance traveled due to the sensor position optimization. This function is introduced in both the processes of relocating sensors for detection and also coverage rate optimization. The *Root Mean Square* (RMS) distance is used to express the mobility cost,

$$d_{\text{RMS}} = \sqrt{\frac{1}{|\mathcal{S}_{mob}|} \sum_{j \in \mathcal{S}_{mob}} {d^{(j)}}^2},$$

where $\mathcal{S}_{mob}$ is the set of sensors being mobilized, and $d^{(j)}$ is the Euclidean distance traveled by the jth sensor in $\mathcal{S}_{mob}$. By normalizing the RMS distance the fitness function can be expressed as

$$F_{\text{dist}} = \frac{d_{\text{RMS}}}{N_x \times N_y}. \tag{3.24}$$

3.5.4 Objective Function

To optimize the sensor locations inside the ROI after attraction, either the GDOP or K-coverage fitness functions can be used,

$$X^*_{\mathcal{S}_{\text{ROI}}} = \min_{X \in (\mathcal{G}_{\text{ROI}} \setminus X_{\text{VTP}})} \left(\omega * F_{\text{ROI}} + (1-\omega) * F_{\text{dist}}\right), \tag{3.25}$$

where $X^*_{\mathcal{S}_{\text{ROI}}}$ is optimized sensor coordinates in the ROI. $\mathcal{G}_{\text{ROI}}$ is the set of grid points in the ROI. Here $F_{\text{ROI}} = \{F_{\text{GDOP}}, F_{\text{Kcov}}\}$. Also, $\omega \in [0, 1]$ is a chosen weight used to combine the fitness functions. The objective function to optimize sensor location outside the ROI can be similarly expressed,

$$X^*_{\mathcal{S}_{\text{out}}} = \min_{X \in \mathcal{G}_{\text{out}}} \left(\omega * F_{\text{uncov}} + (1-\omega) * F_{\text{dist}}\right), \tag{3.26}$$

where $X^*_{\mathcal{S}_{\text{out}}}$ is the optimized locations of the sensors outside the ROIs. Also, $\mathcal{G}_{\text{out}}$ is the set of grid points outside the ROI and within the largest traverse-able region.

3.6 Simulation and Results

The performance of the presented system and algorithms are assessed through a set of experiments. Each experiment examines the problem from a different aspect, with

Table 3.2
Experiments Description

	Description
Experiment 1	Explore which fitness function offers best optimization to minimize the MS position error. Investigate the impact of the node selection algorithm on the MS position error.
Experiment 2	Examine the impact of relocation on the field coverage.
Experiment 3	Investigate the relocation efficiency in terms of mobility cost.

the purpose of achieving a clear picture on the overall behaviour; see Table 3.2. The values of common parameters used throughout the simulations are listed in Table 3.3. All the simulations and results presented in this work are generated using Superior, a high performance computing cluster at Michigan Technological University. No parallel computing has been used in running the simulations. The CPU used in the Superior cluster is the *Intel Sandy Bridge E5-2670 2.60 GHz*.

Experiment 1

The purpose of this experiment is to investigate the different fitness functions proposed to optimize sensor locations and their impact on the MS position error. The comparison is done at different attraction ratios $R_{\text{attraction}}$. Table 3.4 shows the MS position error for this experiment; it is clear that the process of sensor node relocation always offers an improvement over the the initial sensor deployment. This is also

Table 3.3
Values of Common Parameters Used in Simulations

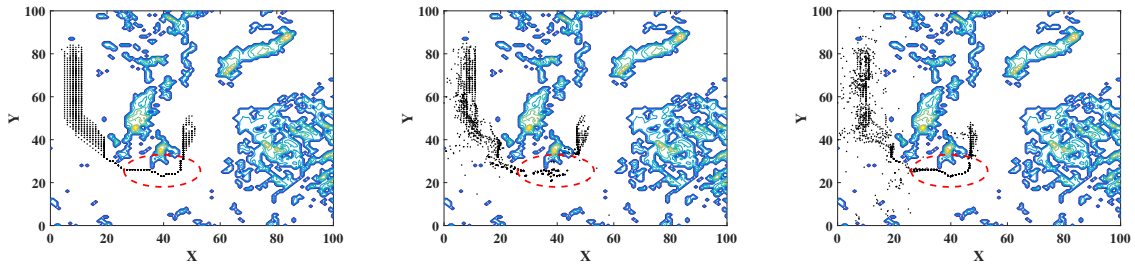
	Parameter	Value(s)
Problem Setup	$N_x \times N_y$	100×100 km
	M	3
	N	100
	$R_{attraction}$	$\{0.20, 0.25, 0.30\}$
	R_s	$\{5, 8, 11\}$
	r_e	0.6
	p_{th}	0.85
	α	0.5
	ξ	0.5
	ω	0.8
GA	p_c	0.75
	p_m	0.001
	N_{pop}	30
	$g_{\max}$	100

observed in terms of missing estimates due to coverage holes, where Fig. 3.5 shows how this problem is remedied. Going back to Table 3.4, the K-Coverage measure as a fitness function seems to offer a better performance. To make sure that the improvement is due to optimizing sensor locations and not only to increasing their density within the ROI, Fig. 3.6 compares three cases. The first is the optimization of sensor locations via the GDOP fitness, the second case uses the K-coverage fitness, and the last case represents the attraction of sensor nodes into the ROI with locations following a uniform random distribution. The comparison is performed at incrementing values of $R_{\text{attraction}}$ and for $R_{\text{GNS}} = [5, 8, 11]$. It is observed that the global node selection algorithm does not take advantage of the GDOP optimization especially at lower R_{GNS} values; where the K-coverage and uniform distribution show a better performance. As R_{GNS} increases, GDOP shows an improved performance

Table 3.4
Comparing the MS position error (meter) under different optimization methods. $R_{\text{GNS}} = 8$.

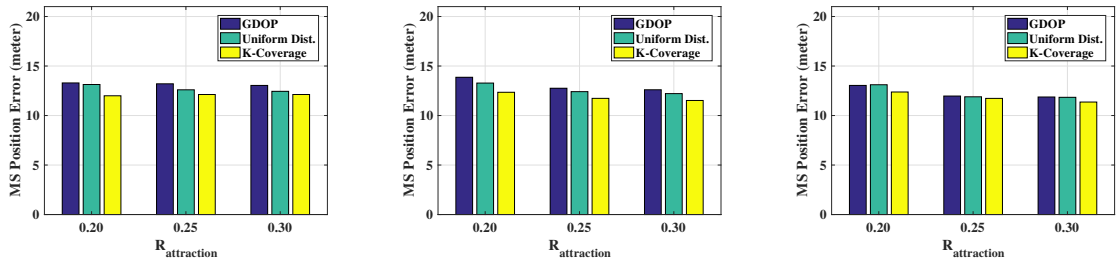
	$R_{\text{attraction}}\%$		
	20	**25**	**30**
Initial	14.7	14.7	14.7
GDOP	13.9	12.8	12.6
K-Coverage	12.4	11.7	11.5

that is comparable with the uniform distribution case, but they both lag behind that of the K-coverage. The improvement in the GDOP method is due to a higher $|\mathcal{S}_a|$ —on average— at larger R_{GNS}, see equation (3.19). It would be interesting in future work to investigate the performance of the GDOP method for higher values of M in a dense network. The results in this experiment are interesting in the sense that a simple uniform distribution of sensor locations leads to a performance that is comparable —if not better— than a more complex methods; namely, the GDOP and K-coverage. So the simpler and computationally more affordable methods seem to offer a better performance.



(a) Ground-truth tracks (b) Before relocation (c) After relocation

Figure 3.5: Impact of relocation on missing location estimates.



(a) $R_{\text{GNS}} = 5$ (b) $R_{\text{GNS}} = 8$ (c) $R_{\text{GNS}} = 11$

Figure 3.6: The impact of optimizing the sensors locations inside the ROI as compared to only attracting the sensor nodes. Comparison is performed for $R_{\text{attraction}} = [0.2, 0.25, 0.30]$.

Experiment 2

It is expected that after relocating the nodes to the ROI, coverage holes will form in the rest of the field. This can result in missed detections of targets that appear in a different part of the field. To reduce the impact of this problem, sensors that remained outside the ROI are relocated to mend the coverage holes. This is performed using a GA with the objective function presented in Section 3.5.4. Figure 3.7 shows the field coverage rate for three cases: i) initial sensor deployment, ii) after relocation to the ROI (with no coverage optimization outside the ROI), and iii) after relocation to the ROI and with coverage optimization. It is clear that relocating nodes to the ROI decreases the coverage rate. It is also observed that optimizing the coverage rate outside the ROI reduces the coverage holes problem. From Fig. 3.7, it seems that small to no improvement is achieved at $R_{\text{attraction}} = [0.25, 0.30]$. This is only due to the fact that at those attraction rates, most—if not all—of the sensors in the field

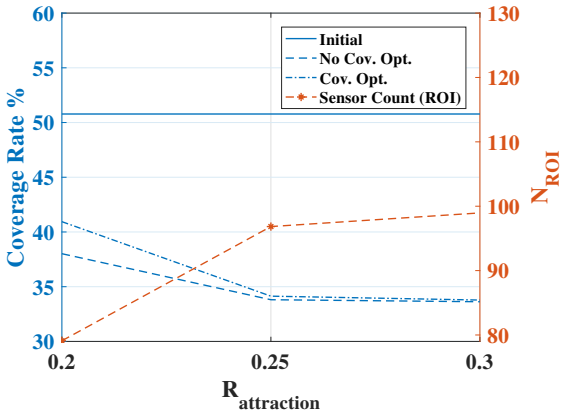


Figure 3.7: Effect of sensor nodes relocation on field coverage rate. $\{F_{\text{ROI}} \leftarrow F_{\text{Kcov}}, R_{\text{GNS}} = 8\}$.

Table 3.5
Impact of R_{GNS} on the coverage rate. The case of initial deployment is considered for illustration purpose.

	R_{GNS}		
	5	**8**	**11**
$R_{\text{COV}}\%$	25	50.8	69

have been relocated to the ROI; see N_{ROI} in Fig. 3.7. From experiment 1 and the results seen here, using either the K-Coverage or the uniform distribution with a low attraction rate offers a reasonable balance between coverage and positioning accuracy. Clearly, the higher the value of R_{GNS} the higher the coverage rate is, Table 3.5.

Experiment 3

This experiment explores the mobility cost in terms of the average traveled distance by all the nodes included in the relocation process. As illustrated in Table 3.6, the traveled distance increases as the number of sensors in the ROI increases. Moreover,

Table 3.6
Average cumulative distance (km) traveled by sensor nodes due to relocation
(inside ROI) and coverage rate optimization (outside ROI).
$\{F_{\text{ROI}} \leftarrow F_{\text{Kcov}}, R_{\text{GNS}} = 8\}$.

		$R_{\text{attraction}}\%$		
		20	**25**	**30**
Inside ROI	d_{RMS} opt.	2577.7	3150.2	3233.7
	No d_{RMS} opt.	2792.8	3511.6	3464
Outside ROI	d_{RMS} opt.	585.1	89.4	35.9
	No d_{RMS} opt.	900.3	186.2	55.8

Table 3.6 shows the gain due to including distance traveled, d_{RMS}, in the objective function. For both the inside and outside the ROI relocation, considering d_{RMS} in the optimization process offers a reduction in the traveled distance of about $7\% - 52\%$. Again, at $R_{\text{attraction}} = [0.25, 0.30]$ minimal or no relocation happens outside the ROI for coverage rate optimization; hence, the sensors experience a relatively smaller traveling distance.

3.7 Conclusion and Discussion

This work investigated the problem of relocating sensor nodes in a WSN for improved target localization and tracking. A system of three phases was introduced to address the questions of where and how to relocate, and how to optimize the location of the sensor nodes with respect to target detection and tracking. Using the initial deployment of the WSN, a set of target locations was established. This set was used

to generate a ROI based on a distribution function estimated by a kernel density estimator. The proposed system increased the sensor density inside the ROI and explored the possibility of optimizing their locations for improved target tracking. Two fitness functions were used to optimize the sensor locations inside the ROI: GDOP and K-coverage. For any nodes that remained outside the ROI, a relocation process was proposed to mend the coverage holes. Results show how the relocation of nodes to the ROI always offers an improvement in terms of reducing the mean-square position error. A comparison between the case of optimizing sensor locations in the ROI and that of just relocating nodes to random locations, was held. The K-coverage showed better performance at all sensor detection range R_{GNS} values, while the GDOP fell behind especially at lower R_{GNS}. Moreover, the uniform random relocation offered a good performance falling shortly behind the K-coverage. In general, increasing the value of R_{GNS} improves the performance in terms of mean-square position error and coverage rate, but has the effect of increasing the search space of the GNS algorithm. If the complexity added by the GNS algorithm at higher R_{GNS} values is not an issue, then either the K-Coverage or the uniform random relocation can be chosen.

Chapter 4

Dynamic Greedy Scheduling for Sparse Sensing in Hybrid Sensor Networks

4.1 Introduction

In many real world scenarios there exist various natural or man-made events that require monitoring with a high level of precision and accuracy [49, 50, 51]. This can range from wild fires and air pollution to contamination of water with nuclear deposits. Achieving high precision requires the use of a network of *high precision sensor* (HPS)

nodes to acquire measurements that allow one to accurately infer the phenomenon of interest. HPS nodes tend to be both expensive and power demanding, which could render such a network to be economically unfeasible [52]. Hence, in practice lower number of nodes are used, forming a sparsely distributed network. Introducing wireless *low precision sensor* (LPS) nodes can help mitigate this problem by filling the gaps in a sparse HPS network, thus rendering the network as a hybrid between two types of sensor nodes. An LPS nodes are likely to be cheaper, easy to deploy, and also consume less power. On the other hand, LPS nodes are typically more prone to failure, mainly due to their dependency on a battery for power. To extend the lifetime of the whole network, it would be crucial to optimize its power consumption, especially for wireless nodes in the hybrid network. Three main areas are addressed in the literature to optimize the power consumption in a *wireless sensor network* (WSN): (1) data processing, (2) communication, and (3) sensing. For WSNs, it is more common to tackle the first two areas [53, 54], but for optimal power consumption the sensing part should also be taken into consideration. While compressing data after sampling is a common practice in a WSN, *compressed sensing* (CS) alters the process by integrating compression into the sensing step, i.e., the field is sensed less frequently. Sampling means fewer nodes are actively sensing the *region of interest* (ROI), which means lower power consumption and a longer life span for the WSN. Let $\mathbf{x} \in R^N$ be a discrete form of the data to be measured, and $\mathbf{y} \in R^M$ be a vector of the data

sensed by the WSN, where $M \ll N$, then

$$\mathbf{y} = \mathbf{\Phi}\mathbf{x} + \mathbf{w}, \tag{4.1}$$

where $\mathbf{w}$ is a noise vector. CS tries to find a measurement matrix $\mathbf{\Phi} \in R^{M \times N}$ such that $\mathbf{y}$ attains sufficient information from the field to achieve an acceptable reconstruction of $\mathbf{x}$. This only work reliably when $\mathbf{x}$ is sparse in a given dictionary Γ,

$$\mathbf{x} = \Gamma\mathbf{a}, \tag{4.2}$$

where $\mathbf{a} \in R^N$ is a sparse representation of $\mathbf{x}$, such that $\|\mathbf{a}\|_0 \ll N$ [55]. There are several approaches to solve the minimization $\mathbf{x}$; ℓ_1, greedy algorithms, and combinatorial algorithms. The ℓ_1 based minimization is an approximation for that with ℓ_0; the ℓ_0 minimization is non-convex and can be shown to be NP-hard to solve [56]. The ℓ_1 minimization in the presence of noise has the following expression [57],

$$\min_{\mathbf{a} \in R^N} \|\mathbf{a}\|_1, \quad \text{s.t. } \|\mathbf{y} - \mathbf{\Phi}\Gamma\mathbf{a}\|_2 \leq \xi_{\mathbf{w}}, \tag{4.3}$$

where $\xi_{\mathbf{w}}$ is the noise energy. The problem presented at (4.3) is convex and can be solved by many optimization methods [58]. Two of the greedy algorithms are *orthogonal matching pursuit (OMP)* [59] and *iterative hard thresholding* (IHT) [60]. OMP and IHT can achieve similar guarantees as the ℓ_1 minimization for recovering $\mathbf{x}$.

Now, due to the non-adaptive nature of the sensing schedule captured in $\mathbf{\Phi}$, which is usually either random or fixed, the use of CS for sparse sensing in a WSN might not be the best practice [61]. Also, the sparsity of the dictionary Γ changes as $\mathbf{x}$ varies with time, which in effect can fail to fulfill the CS requirements [61, 62]. Zichong Chen et al. [61] presented a distributed algorithm to provide a near-optimal spatio-temporal measurement schedule for a group of stationary sensing nodes in a WSN. There, the authors considered a decomposition of $\mathbf{x}$ at time t,

$$\mathbf{x} = \mathbf{\Psi}^{(t)}\boldsymbol{\nu}, \tag{4.4}$$

where $\mathbf{\Psi}^{(t)} \in R^{N\times K}$ is the signal approximation model at time t, and $\boldsymbol{\nu} \in R^K$ is a low dimensional representation of $\mathbf{x}$, such that $K \ll N$. The reconstruction of $\mathbf{x}$ requires the knowledge of $\mathbf{\Psi}^{(t)}$, which can be learned over time from previous measurements. Through this process, the measurements schedule $\mathbf{\Phi}^{(t)}$ can also be learned.

Our work is an extension of the adaptive scheduling algorithm presented by Chen [61]. For a hybrid WSN that is composed of both HPS and LPS nodes, our contribution aims at investigating the use of sparse sensing algorithms to provide measurement scheduling that is both dynamic and accurate, i.e., achieves a low reconstruction error. The process of scheduling is considered dynamic and adaptive in the sense that it adapts to the ever changing attributes of the physical field. As a case study, the concentration of *particulate matter* with an aerodynamic diameter of less than $10\mu m$

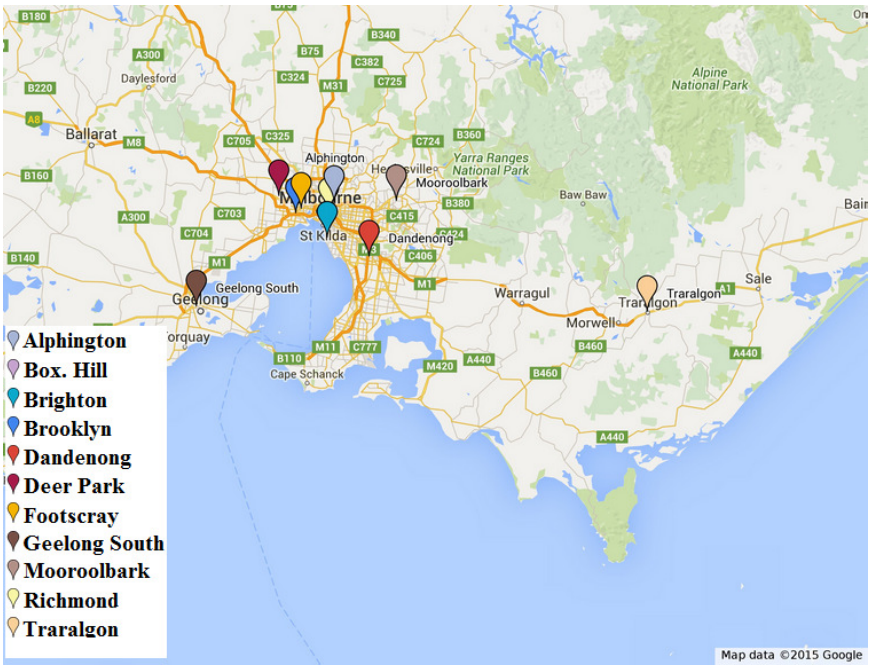


Figure 4.1: Sensing stations in the Melbourne, Australia area / Google Maps.

(PM10) is used as the dataset of interest to assess the performance of the presented system. This dataset was provided by the *Environment Protection Authority* Victoria (EPA) [63]. It offers hourly PM10 measurements by eleven stations distributed in the Melbourne area, see Fig. 4.1. These stations are considered HPS nodes; their measurements are used to develop a model for generating simulated ground-truth PM10 data. This will help in two ways: (1) in filling the gap in measurements due to the sparsely located HPS nodes, and (2) in studying the impact of using LPS nodes to achieve a better estimation of the PM10 concentrations in the region.

In contrast to what is usually seen in the CS literature, this work adopts a sparse sensing approach that is dynamic and provides a frame work of scheduling sensor nodes

for measurement based on their information content at a given point in time. This work address problems related WSN deployment feasibility and network resilience to node failures that were not dealt with in [61, 64], and in that context here are the main contribution seen in this chapter,

† The unique hybrid structure of the network, where the impact of utilizing LPSs is examined;

† Investigating the trade-off between network deployment feasibility and measurement reconstruction accuracy;

† Exploring the resilience of the greedy measurement-scheduling algorithms to sensor node failures.

† A comprehensive discussion on the generation of simulated ground-truth data, over which the system is applied.

Table 4.1 provides a summery of the important notations used here. The rest of this chapter is arranged as follows. In Section 4.2, we present a more detailed description of the problem at hand, with a preview on the synthetic ground-truth data generation. Section 4.3 introduces the four steps that comprise the dynamic measurement scheduling and presents two greedy algorithms as a solution. Results and discussion are presented in Section 4.4. Finally, the conclusion and future work are found in Section 4.5.

Table 4.1
Acronyms and Notation

Term	Definition	Term	Definition
$\mathbf{x}$	ground-truth data of N sensors	$\mathbf{\Phi}$	Measurement Matrix
$\mathbf{w}$	Noise vector	$\mathbf{y}$	Measurements vector
$\mathbf{a}$	Sparse form of $\mathbf{x}$	$\xi_{\mathbf{w}}$	Noise energy
$\mathbf{\Psi}$	Signal approximation model	$\boldsymbol{\nu}$	Low dimensional form of $\mathbf{x}$
$\boldsymbol{\tau}$	Measurement pattern	N	Number of sensors
M	Number of samples (measurements)	$C_s(d)$	Spatial lag covariance model
$C_t(\Delta t)$	Time lag covariance model	ϵ_a	Approximation error
$\mathbf{C_x}$	Covariance of $\mathbf{x}$	$\bar{\mathbf{x}}$	Mean of $\mathbf{x}$
$\hat{\mathbf{x}}$	Approximation of $\mathbf{x}$	$\tilde{\mathbf{x}}$	Reconstruction of $\mathbf{x}$
ϵ	Reconstruction error	FP(.)	Frame potential measure
WSN	Wireless sensor network	HPS	High precision sensor
LPS	Low precision sensor	ROI	Region of interest
CS	Compressed sensing	PM	Particulate matter
OMP	Orthogonal matching pursuit	IHT	Iterative hard thresholding
FFT	Fast Fourier transform	WSS	Wide-sense stationary

4.2 Problem Setup

In a WSN with N sensing nodes, a given node has its spatio-temporal coordinates expressed as $\mathbf{P} = \{\mathbf{s}, t\}$, where $\mathbf{s} \in R^2$ is the spatial coordinate and t is time. Let $\mathcal{Z}^{(t)} = \{Z_1^{(t)}, Z_2^{(t)}, ..., Z_N^{(t)}\}$ be the set of their measurements at time t. Also, let N_{H} and N_{L} be the number of HPS nodes and LPS nodes, respectively. An HPS is a sensing node that acquires measurements from the field with a high *signal-to-noise ratio* (SNR); these sensors are considered to be expensive, hence their count is usually lower than the relatively inexpensive, but less accurate, LPSs. Any given sensor is assumed to be in an either an active or sleep state. Nodes that are in the sleep mode

are not capable of acquiring measurements. Hence, let $N_{\mathrm{H}}^{(\mathrm{a})}$ and $N_{\mathrm{L}}^{(\mathrm{a})}$ be the number of active sensors for HPS and LPS nodes, respectively. Also, $N_{\mathrm{H}}^{(\mathrm{s})}$ and $N_{\mathrm{L}}^{(\mathrm{s})}$ is the number of sleeping sensors for HPS and LPS nodes, respectively.

Even though the data to be measured is continuous in space and time, we assume that we are dealing with data sampled at minimum of the Nyquist rate. In this work, data is processed in frames of length $N = N_{\mathrm{L}} + N_{\mathrm{H}}$, representing the measurements from all sensing nodes. Based on previous measured data from the physical field, the proposed algorithm will select $M = N_{\mathrm{H}}^{(\mathrm{a})} + N_{\mathrm{L}}^{(\mathrm{a})}$ sensors, from the N possible nodes, to be sampled at $t+1$. More details on this process are provided in Section 4.3. Indices of the selected M nodes are stored in $\boldsymbol{\tau}^{(t+1)}$, which is referred to as the measurement (or sampling) pattern. The measurement schedule $\boldsymbol{\Phi}^{(t+1)} \in \{0,1\}^{M \times N}$ can be completely determined from $\boldsymbol{\tau}^{(t+1)}$ as follows,

$$\phi_{i,j}^{(t+1)} = \begin{cases} 1, & j = \tau_i^{(t+1)} \\ \\ 0, & \text{else} \end{cases}, \tag{4.5}$$

where $\phi_{i,j}^{(t+1)}$ is the element in the ith row (ith index in $\boldsymbol{\tau}$) and jth column (jth of N sensors) of $\boldsymbol{\Phi}^{(t+1)}$. From (4.5) it is observed that $\boldsymbol{\Phi}^{(t+1)}$ has a maximum of one non-zero in each column, and exactly a single non-zero element in each row.

4.2.1 Simulated Ground-Truth Data

Rajasegarar et al. [64] used two methods for noise generation based on fitted covariance modules. The first relies on the *fast fourier transform* (FFT) and the other on Cholesky factorization. For generating the synthetic PM10 ground-truth data, we will follow an adaptation of the FFT method, assuming that sensors are located on a grid in the field of interest. The covariance model is assumed to be *wide-sense stationary* (WSS) and isotropic. There are many models that can be used for this purpose. For illustration purposes, let us consider the exponential model,

$$C(d) = \alpha e^{-d/\sigma},$$

where α and σ are the model parameters and d is the spatial lag. As mentioned previously, the FFT method considers the sensor nodes to be positioned on a grid. Let a sensor be positioned at $\mathbf{s}_{i,j} = (x_i, y_j)$. The spatial covariance matrix is expressed as $\mathbf{C} = [C(d_{i,j})]_{m \times n}$, where $d_{i,j} = \parallel \mathbf{s}_{i,j} - \bar{\mathbf{s}} \parallel_2$, and $\bar{\mathbf{s}}$ is the grid center. The FFT model generates the synthetic data as shown in Fig. 4.2, where $\mathbf{W} \sim U[0,1]$ is an $m \times n$ uniform noise matrix, $\bigodot$ is the Hadamard product, and $\mathbf{Z}$ is the ground-truth data matrix. By taking the Fourier transform of the covariance matrix $\mathbf{C}$, the power spectral density (PSD) is evaluated. To apply the statistics of the of the measured dataset to the noise (i.e., synthetic data), we modify the magnitude of the complex

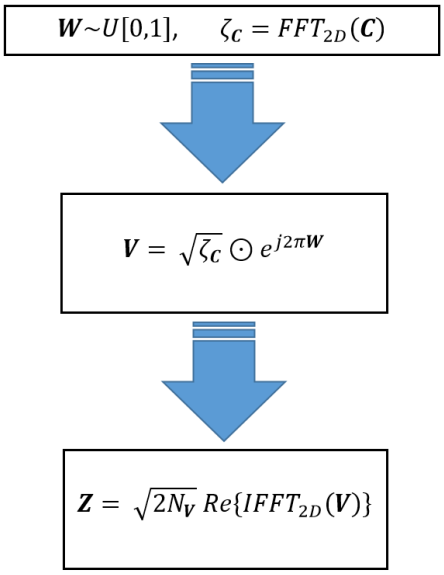


Figure 4.2: Flow diagram of the FFT model for ground-truth data generation.

numbers in $e^{j2\pi\mathbf{W}}$ by multiplying with $\sqrt{\zeta_{\mathbf{C}}}$. The use of the square-root is to preserve the covariance of $\mathbf{Z}$ to the PSD $\zeta_{\mathbf{C}}$. The final step brings $\mathbf{V}$ back from the frequency domain. The multiplication with $\sqrt{2N_{\mathbf{V}}}$ counteracts the implicit multiplication by $1/2$ due to the $Re\{.\}$ operation, and by $1/N_{\mathbf{V}}$ due to the inverse FFT definition, where $N_{\mathbf{V}}$ is the number of coefficients in $\mathbf{C}$.

Since we are modeling data for sensors located in a spatio-temporal space, the model should capture both spatial and temporal statistics. Therefore, the spatio-temporal covariance model is expressed as,

$$C(d, \Delta t) = C_s(d) \times C_t(\Delta t),$$

where $C_s(d)$ and $C_t(\Delta t)$ are the isotropic WSS covariance models for the spatial lag d and time lag Δt, respectively. It is important to note that the spatio-temporal covariance should have the same α parameter as in $C_s(d)$ and $C_t(\Delta t)$, see Table 4.2. Hence, $C(d = 0, \Delta t = 0) = C_s(d = 0) = C_t(\Delta t = 0)$. This preserves energy.

For the synthetic data to reflect the measured dataset, it is important to estimate the models $C_s(d)$ and $C_t(\Delta t)$ that best fit their spatial and time lag covariances of the measured data. This can be achieved by first computing the spatial and temporal variograms from the measured dataset [64, 65, 66]. The spatial variogram is expressed as

$$\hat{C}_s(d) = \sum_{t=1}^{T} \sum_{\forall i,j} \frac{(Z_i^{(t)} - \mu_Z)(Z_j^{(t)} - \mu_Z)}{T n_d}, \tag{4.6}$$

where n_d is the number of covariance terms for each spatial lag d, and μ_Z is the mean of the measurements. The temporal variogram is

$$\hat{C}_t(\Delta t) = \sum_{m=1}^{N_s} \sum_{\forall i,j} \frac{(Z_m^{(t_i)} - \mu_Z)(Z_m^{(t_j)} - \mu_Z)}{N_s n_{\Delta t}}, \tag{4.7}$$

where $n_{\Delta t}$ is the number of terms in the second summation corresponding to the temporal lag Δt, and N_s is the number of sensors. From a set of possible covariance models, a best fit is chosen based on minimizing the squared error,

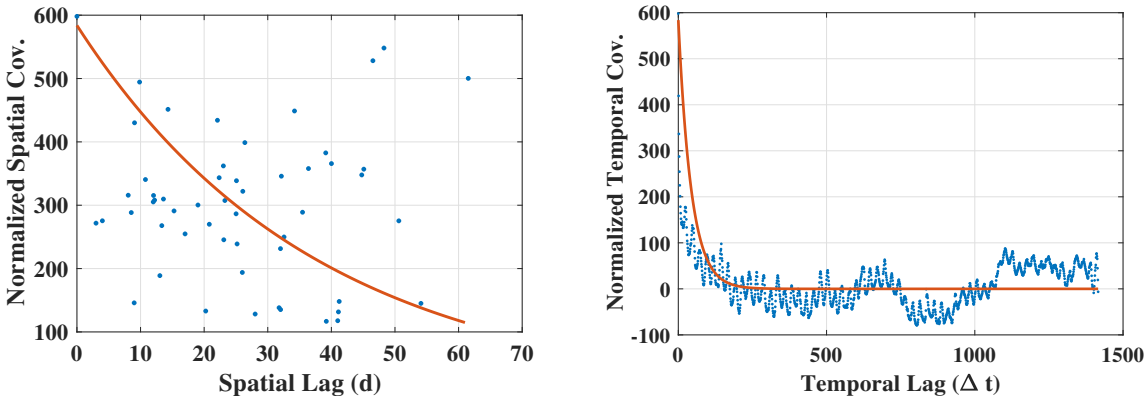
$$\epsilon_u(\mathcal{M}) = \sum_g n_g^2 \left(\hat{C}(g) - C_u(g, \mathcal{M}) \right)^2, \tag{4.8}$$

Table 4.2
List of covariance models

Name	Model Parameters ($\mathcal{M}$)	Formula
Exponential	$\{\alpha, \sigma\}$	$C_1(q, \mathcal{M}) = \alpha e^{-q/\sigma}$
Gaussian	$\{\alpha, \sigma\}$	$C_2(q, \mathcal{M}) = \alpha e^{-q^2/\sigma^2}$
Cosine Hole	$\{\alpha, \sigma\}$	$C_3(q, \mathcal{M}) = \alpha \cos(\pi q/\sigma)$
Sinc	$\{\alpha, \sigma\}$	$C_4(q, \mathcal{M}) = \alpha \sin(\pi q/\sigma)/(\pi q/\sigma)$
Nugget	α	$C_5(q, \mathcal{M}) = \alpha, C_5(q, \mathcal{M}) = 0, q > 0$
Spherical	$\{\alpha, \sigma\}$	$C_6(q, \mathcal{M}) = \alpha(1 - [3q/2\sigma - 1/2(q/\sigma)^3])$
Mexican Hat	$\{\alpha, \sigma\}$	$C_7(q, \mathcal{M}) = \alpha(1 - \sigma q^2)e^{q^2/\sigma^2}$

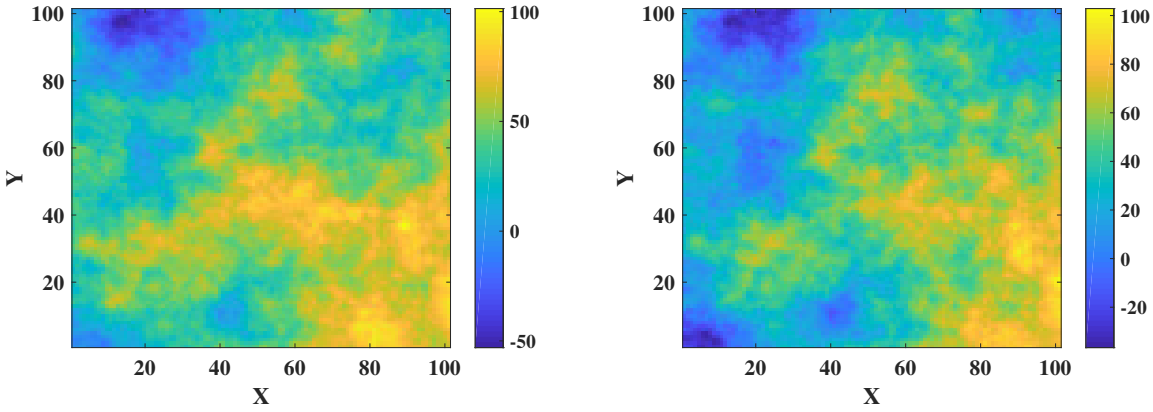
where $\mathcal{M}$ is the set of model parameters and g represents all possible time or spatial lags. $C_u(g, \mathcal{M})$ is one of a set of possible covariance models, shown in Table 4.2. Due to the fact that the sensors are sparsely located in the field, there will be gaps present in the spatial variogram. Therefore, it would be possible to interpolate $\hat{C}_s(d)$ before fitting model, keeping in mind that a result of this would be using $n_d = 1$ at (4.6) due to the fitting over a uniform grid. However, for $\Delta t = t_i - t_j$ we have $n_{\Delta t} = \{T, T-1, ...\}$.

Figure 4.3 shows the fit of the best model, exponential, to both temporal and spatial variograms. The parameters for the spatial models are $\mathcal{M}_s = \{584, 37.5\}$, and for the temporal model $\mathcal{M}_t = \{270, 44\}$. Since both the spatial and temporal variograms start at the same point (around 600), both models get the same α parameter. Observing that $\mathcal{M}_s$ has an α value that is more true to the spatial and temporal variograms, both models get an α parameter of $\alpha = 584$. Using the fitted models, the FFT method is used to generate the ground-truth synthetic data, as shown in Fig. 4.4. From Fig. 4.5, it is clear that the sample-based covariance (either spatial



(a) Spatial (b) Temporal

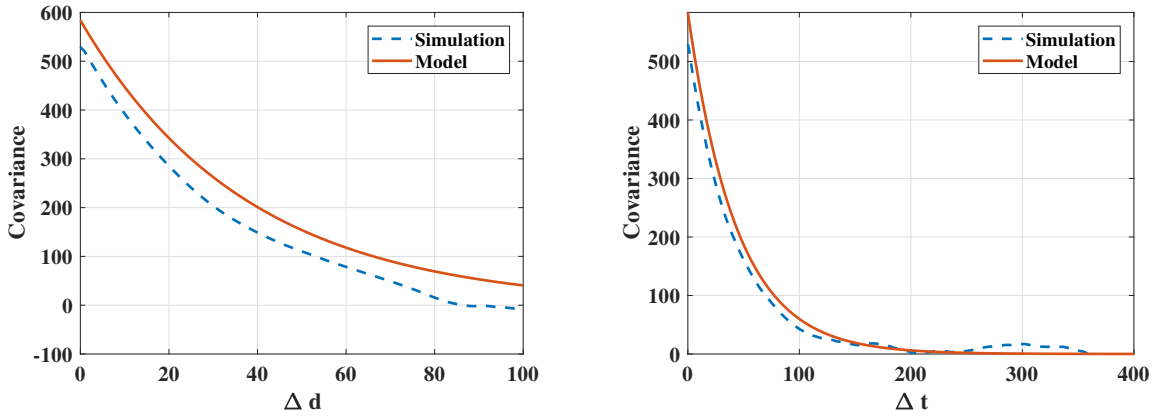
Figure 4.3: Fitting a model for both spatial and temporal variograms.



(a) (b)

Figure 4.4: Simulated ground-truth data for two time slices. The FFT method was used with the exponential model for both spatial and temporal covariance models.

or temporal) of the simulated data follows the trend of the covariance models that are based on the measured dataset. As the number of simulated data grows, the covariance models, as estimated from the simulated data, match the measured dataset even closer.



(a) Spatial covariance (b) Temporal covariance

Figure 4.5: Comparison between the covariance model based on the measured data and the sample-based covariance of the simulated ground-truth data.

4.3 Dynamic Measurement Scheduling

We now provide details regarding the procedure of dynamically assigning a measurement schedule to the sensing nodes, both HPSs and LPSs. The four steps of this procedure are illustrated in Fig. 4.6.

4.3.1 Step 1: Measurement Acquisition

In this step, the sensors get the measurements according to the schedule $\mathbf{\Phi}^{(t)}$. This means that there will be $N_{\mathrm{H}}^{(\mathrm{a})} + N_{\mathrm{L}}^{(\mathrm{a})}$ active HPS and LPS nodes out of N nodes taking measurements.

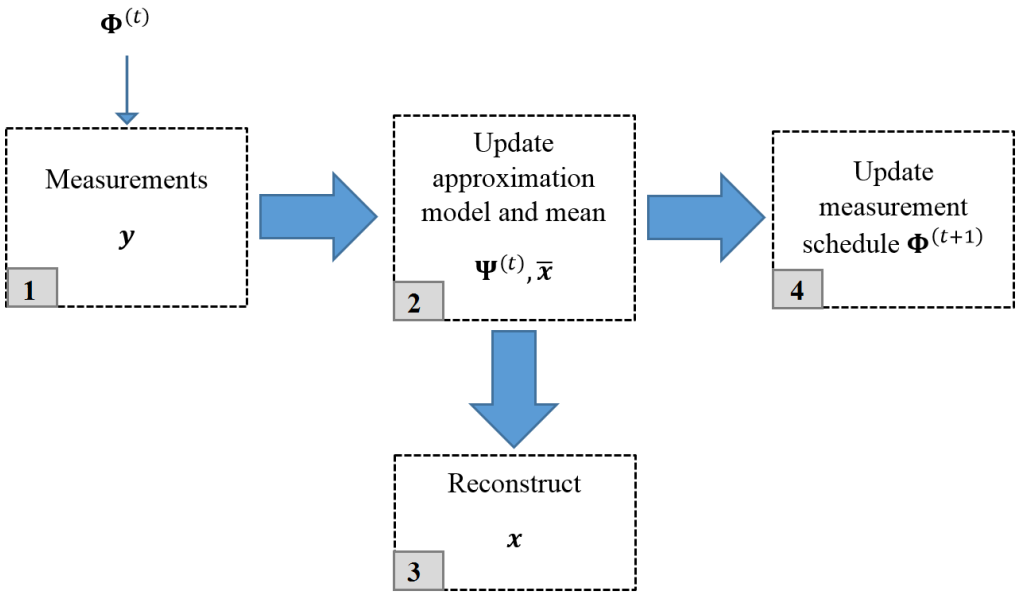


Figure 4.6: Flow diagram of the dynamic measurement scheduling.

4.3.2 Step 2: Updating $\mathbf{\Psi}^{(t)}$ and $\bar{\mathbf{x}}$

The optimal model $\mathbf{\Psi}^{(t)}$ is that which minimizes the approximation error ϵ_a,

$$\epsilon_a = \frac{1}{\sqrt{N}} E[\| \mathbf{x} - \hat{\mathbf{x}} \|_2], \tag{4.9}$$

where $\hat{\mathbf{x}} = \mathbf{\Psi}^{(t)}\boldsymbol{\nu} + \bar{\mathbf{x}}$ is the approximation of the field data $\mathbf{x}$, and $\bar{\mathbf{x}}$ is its mean. Estimating $\mathbf{\Psi}^{(t)}$ can be achieved by analyzing previous field measurements. Since $\mathbf{\Psi}^{(t)} \in R^{N \times K}$ is a K-dimensional subspace, where $K \ll N$, this can be viewed as a dimensionality reduction problem; this is addressed using *principle component analysis* (PCA).

To proceed with PCA, the covariance matrix $\mathbf{C_x}$ is required. This can be challenging for two main reasons: (1) we only have M out of N measurements, and (2) $\mathbf{C_x}$ is changing with time. Exploiting the fact that measurements are potentially sampled from different locations at different time instances, enough information can be acquired for estimating $\mathbf{C_x}$ and hence updating $\mathbf{\Psi}^{(t)}$ and $\bar{\mathbf{x}}$. To that end, two methods were introduced in [61]. The first is based on a first-in first-out (FIFO) buffer for storing and analyzing L previous measurement frames, while the second is based on incremental PCA (IPCA). Here, the later method is adopted due to its memory efficiency and better performance in terms of reconstructing $\mathbf{x}$. The FIFO method is only used to initialize the IPCA algorithm. Both methods rely on an interpolated version of the measurements $\mathbf{y}$; due to the spatial nature of the measurements, a cubic spatial interpolation is used where Matlab's '*griddata*' was employed for this purpose.

4.3.3 Step 3: Reconstruction

The field reconstruction is based on the measurement approximation model $\hat{\mathbf{x}}$

$$\tilde{\mathbf{x}} = \mathbf{\Psi}^{(t)}\boldsymbol{\nu} + \bar{\mathbf{x}}, \tag{4.10}$$

where $\tilde{\mathbf{x}}$ is the reconstruction of the field $\mathbf{x}$ and $\boldsymbol{\nu}$ is the projection coefficient vector reflecting a low dimensional representation of $\mathbf{x}$. Having $\mathbf{\Psi}^{(t)}$ and $\bar{\mathbf{x}}$ estimated, we

need $\boldsymbol{\nu}$ for evaluating $\tilde{\mathbf{x}}$. From (4.1) and using $\hat{\mathbf{x}}$,

$$\begin{aligned}\hat{\mathbf{y}} &= \boldsymbol{\Phi}^{(t)}\hat{\mathbf{x}} + \mathbf{w} \\ &= \boldsymbol{\Phi}^{(t)}(\boldsymbol{\Psi}^{(t)}\boldsymbol{\nu} + \bar{\mathbf{x}}) + \mathbf{w}. \end{aligned} \tag{4.11}$$

For an i.i.d. Gaussian noise $\mathbf{w}$, $\boldsymbol{\nu}$ can be estimated by solving the ordinary least squares (OLS) problem [61, 67],

$$\boldsymbol{\nu}^* = \arg\min_{\boldsymbol{\nu}} \| \mathbf{y} - \hat{\mathbf{y}} \|_2^2. \tag{4.12}$$

The analytic solution for this problem is

$$\boldsymbol{\nu}^* = (\boldsymbol{\Phi}^{(t)}\boldsymbol{\Psi}^{(t)})^{\dagger}(\mathbf{y} - \boldsymbol{\Phi}^{(t)}\bar{\mathbf{x}}), \tag{4.13}$$

where $(.)^{\dagger}$ represents the Moore-Penrose pseudoinverse. At this point a reconstruction of the field is achieved by substituting (4.13) into (4.10) giving

$$\tilde{\mathbf{x}} = \boldsymbol{\Psi}^{(t)}(\boldsymbol{\Phi}^{(t)}\boldsymbol{\Psi}^{(t)})^{\dagger}(y - \boldsymbol{\Phi}^{(t)}\bar{\mathbf{x}}) + \bar{\mathbf{x}}. \tag{4.14}$$

Again assuming an i.i.d. Gaussian noise, the reconstruction error is upper bounded by [61],

$$\epsilon^2 = \frac{1}{N} \parallel \mathbf{x} - \tilde{\mathbf{x}} \parallel_2^2 \leq \frac{1}{\lambda_K} \epsilon_a^2 + \sigma^2 \sum_{k=1}^{K} \frac{1}{\lambda_k}, \tag{4.15}$$

where ϵ is the reconstruction error, λ_i for $1 \leq i \leq K$ are the eigenvalues of $\boldsymbol{\Upsilon} = \tilde{\boldsymbol{\Psi}}^{(t)^T} \tilde{\boldsymbol{\Psi}}^{(t)}$ sorted in a descending order, and $\tilde{\boldsymbol{\Psi}}^{(t)} = \boldsymbol{\Phi}^{(t)} \boldsymbol{\Psi}^{(t)}$. λ_K is the minimum eigenvalue of Υ for the range $1 \leq i \leq K$. σ is the noise variance.

4.3.4 Step 4: Update Measurement Schedule

Here we describe two greedy algorithms for updating $\boldsymbol{\Phi}^{(t)}$ to provide the scheduling update at time $t+1$.

4.3.4.1 Frame Potential

To optimize the measurement schedule the following minimization problem is solved [61],

$$\boldsymbol{\tau}^{(t)^*} = \underset{\boldsymbol{\tau}^{(t)}}{\arg\min} \underset{\lambda_k}{\min} \sum_{k=1}^{K} \frac{1}{\lambda_k}, \tag{4.16}$$

where λ_k are the eigenvalues of $\boldsymbol{\Upsilon}$. However, this problem is considered to be NP-hard [68, 69]. As a proxy for the cost function at (4.16), the frame potential (FP) [71] can be used [61, 70]:

$$\mathrm{FP}(\boldsymbol{\Psi}^{(t)}, \mathcal{A}) = \sum_{i,j \in \mathcal{A}} |\langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j \rangle|^2, \tag{4.17}$$

where $\mathcal{A}$ is the set of sensors indices that are candidates for activation. Also, $\boldsymbol{\psi}_i$ is the ith row of $\boldsymbol{\Psi}^{(t)}$. Minimizing the frame potential in effect addresses the problem at (4.16), and since it is convex, it has a lower computational complexity. Under some conditions over the spectrum of $\boldsymbol{\Psi}^{(t)}$, using the FP as a proxy can achieve a near-optimal solution in terms of the root-mean-square error (RMSE) [70]. Since the FP is a measure of the orthogonality between the rows of $\boldsymbol{\Psi}^{(t)}$, the greedy algorithm selects the rows (sensors locations) that are close to being orthogonal, potentially increasing the information content of the data to be measured [61].

For the evaluation of the measurement schedule for the next time frame, Algorithm 3 is based on a worst-out greedy algorithm. In each iteration, the index of the $\boldsymbol{\Psi}^{(t)}$ row that maximizes the frame potential is removed from the set $\mathcal{A}$. This process is repeated until $|\mathcal{A}| = M$.

Algorithm 3 Frame Potential

procedure INITIALIZATION
  Set $\mathcal{R} = \emptyset$; $\mathcal{R} \leftrightarrow$ removed indices set
  Set $\mathcal{A} = \{1, 2, ..., N\}$
end procedure
First row to remove $\mathcal{R} = \arg\max_{i \in \mathcal{A}} \sum_{i,j \in \mathcal{A}} |\langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j \rangle|^2$
Update $\mathcal{A} = \mathcal{A} \setminus \mathcal{R}$
repeat
  $i^* = \arg\max_{i \in \mathcal{A}} \sum_{i,j \in \mathcal{A}} |\langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j \rangle|^2$
  $\mathcal{R} = \mathcal{R} \cup i^*$
  $\mathcal{A} = \mathcal{A} \setminus i^*$
until $|\mathcal{A}| = M$
$\boldsymbol{\tau}^{(t+1)} \leftarrow \mathcal{A}$

4.3.4.2 Correlation

As seen in [72], the second greedy algorithm is also related to the idea of selecting the nodes that are less related to each other, hence possibly providing more information. Here, the correlation between the $\boldsymbol{\Psi}^{(t)}$ rows are evaluated, and a greedy worst-out method is used to remove the nodes that are highly correlated; see Algorithm 4.

Algorithm 4 Correlation

procedure INITIALIZATION
  Set $\mathcal{R} = \emptyset$; $\mathcal{R} \leftrightarrow$ removed indices set
  Set $\mathcal{A} = \{1, 2, ..., N\}$
end procedure
repeat
  $i^* = \arg\max_{i \in \mathcal{A}} \langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j \rangle$
  $\mathcal{R} = \mathcal{R} \cup i^*$
  $\mathcal{A} = \mathcal{A} \setminus i^*$
  construct $\tilde{\boldsymbol{\Psi}}^{(t)}$ from $\mathcal{A}$
  if rank($\tilde{\boldsymbol{\Psi}}^{(t)}$) $<$ K **then**
    Restore previous $\tilde{\boldsymbol{\Psi}}^{(t)}$
    Break
  end if
until $|\mathcal{A}| = M$
$\boldsymbol{\tau}^{(t+1)} \leftarrow \mathcal{A}$

4.4 Results and Discussion

Table 4.3 offer a brief description of each of the experiments. Some of the codes used in generating the simulations are based on the repository in [73]. Before proceeding

into the experiments, two main parameters need to be defined. The first is RMSE,

$$\text{RMSE} = \frac{1}{\sqrt{N}} \parallel \mathbf{x} - \tilde{\mathbf{x}} \parallel_2 . \tag{4.18}$$

Throughout the experiments a normalized version of the RMSE is used,

$$\text{RMSEN} = \frac{\text{RMSE}}{\parallel \mathbf{x} \parallel_2}. \tag{4.19}$$

The other parameter is SNR, defined as

$$\text{SNR}_{dB} = 10 \log_{10} \frac{\parallel \mathbf{x} \parallel_2^2}{\parallel \mathbf{w} \parallel_2^2}. \tag{4.20}$$

It is worthwhile mentioning that the HPS nodes are assumed to introduce zero (negligible) noise to the measurements, while the LPS nodes are noisy. Hence, varying the SNR value in the simulations reflects the level of noise in the LPS nodes. Unless otherwise mentioned, the common parameters' values used for result generation are those listed in Table 4.4.

Table 4.3
Experiments Description

	Description
Experiment 1	Compares the performance of the various scheduling algorithms in terms of RMSE.
Experiment 2	Investigate the resilience of the scheduling algorithm to node failure.
Experiment 3	Explore the possible gain of increasing the percentage of HPS nodes.
Experiment 4	Investigate the trade-off between increasing the percentage of HPS nodes and the improvement in the RMSE performance.
Experiment 5	Study the impact of having all the HPS nodes in the active state.

Table 4.4
Values of Common Parameters

Parameter	Value
N	169
M	16
K/M	0.9
$\mathrm{P_{HPS}}$	10%

4.4.1 Experiment 1

This experiment compares a set of scheduling algorithms. Two of which provide dynamic scheduling and were presented in Section 4.3. The other two are uniform scheduling and random scheduling. The uniform scheduling selects nodes that are equally spaced from a grid of nodes overlaid over the FOI. The further the nodes are spaced the smaller the sample M. Random scheduling picks nodes based on a uniform random distribution. Figure 4.7(a) shows the normalized RMSEN performance of the

uniform, correlation, and the FP scheduling algorithms. Both the uniform and the FP methods outperform the correlation method, with a slight edge of uniform scheduling over that of FP. One possible reason for the favorable performance of the uniform scheduling lies in the consistent spatial spacing between the nodes, which in turns results in uncorrelated measurements, especially at higher node spacing. One might argue for the superiority of the uniform scheduling, but it must be remembered that it is a static scheduling scheme. With uniform scheduling two main problems arise:

† The same nodes are active all the time, shortening the lifetime of the LPS nodes and hence the network integrity.

† In case of malfunction in a node or set of nodes, the uniform spacing is broken, and as demonstrated in Experiment 2 the advantage in RMSE performance no longer holds.

Based on its RMSE performance and dynamic nature, the FP method is adopted. Figure 4.7(b) compares the random scheduling performance against that of FP. Clearly, random scheduling is the worst performing method.

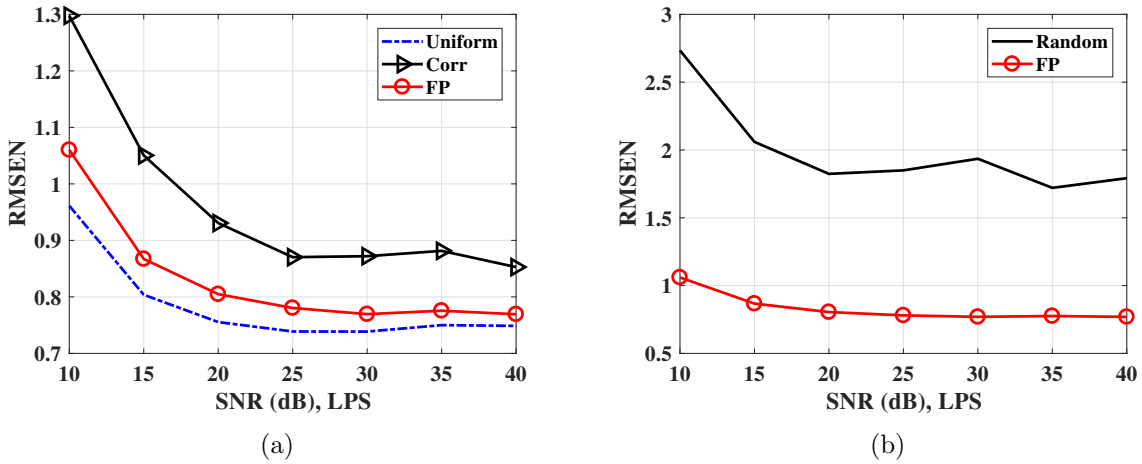


Figure 4.7: RMSE performance among the various scheduling methods.

4.4.2 Experiment 2

Here, a scenario where a random set of the LPS nodes malfunctions is considered. This is a reasonable proposition, mainly due to the reliance of the LPS nodes on a battery as a power source. Both the uniform and the FP methods are considered here. To maintain a fixed sample size M, the uniform scheduling algorithm substitutes failed nodes by the nearest inactive (sleeping) nodes in the grid, while the FP artificially creates a high spatial correlation among failed nodes, encouraging the greedy aspect of the algorithm to discard those nodes from the sample. Figure 4.8 shows that the FP algorithm offers more resilience against node failure. One might wonder about the slight improvement in RMSE performance as the number of failing LPS nodes increases. This is due to the increased chance that the scheduling algorithm will substitute a malfunctioned LPS node with an HPS node.

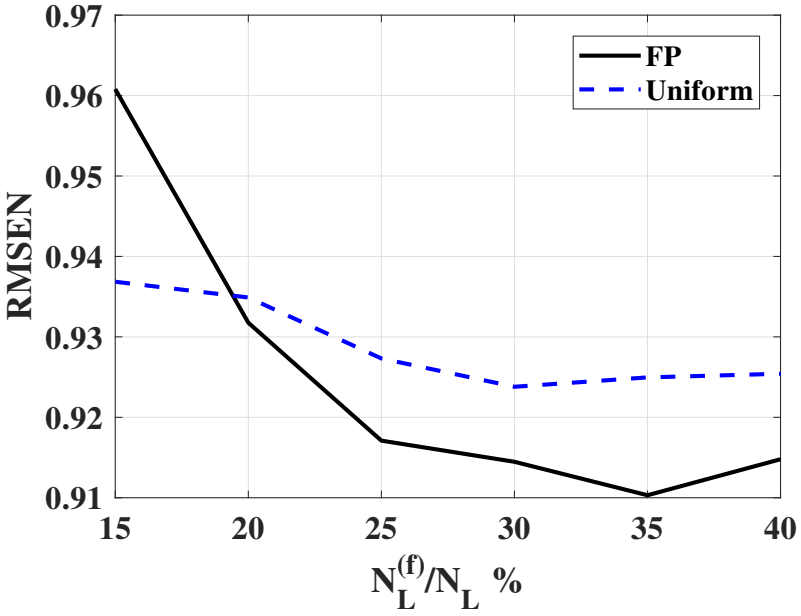
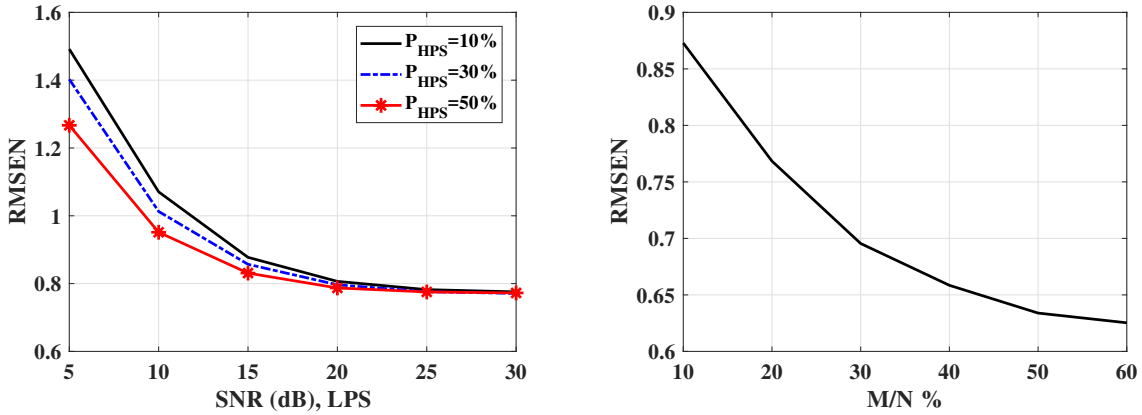


Figure 4.8: Comparing the resilience of both the FP and uniform methods to LPS node failure. The x-axis represents the percentage of failed LPS node of the total LPS node count. SNR = $10dB$.

4.4.3 Experiment 3

This experiment explores the possible performance gain due to increasing increasing the ratio of HPS to LPS nodes. The HPS nodes are advantageous for their high precision measurements, while the LPS nodes are considered noisy. Figure 4.9(a) reveals how increasing the percentage of HPS nodes, P_{HPS}, yields an improved performance mainly lower SNR regimes—i.e., in regions where the LPSs are very noisy. The increase in SNR reflects the use of a higher quality LPS nodes. Hence, as SNR increases, the quality margin between the LPS and HPS nodes shrinks. At SNR $> 20dB$ the LPS nodes virtually turn into HPS nodes, and as a result no performance gain is



(a) (b)

Figure 4.9: Impact of HPS nodes percentage (a) and sample size M (b) on the RMSE performance. The FP algorithm is used for scheduling.

seen from increasing $\mathrm{P_{HPS}}$. It is also observed that after a certain SNR the RMSE performance reaches a plateau. This is a result of the sample size M, and can be mitigated by increasing the value of M at the cost of activating a larger number of nodes; see Fig. 4.9(b).

4.4.4 Experiment 4

This experiment is an extension of Experiment 3, where it was shown how increasing the HPS percentage improved the RMSE performance at a lower SNR regime. But now the question that arises is whether this improvement is worth investing in more HPS nodes. As expected, Fig. 4.10 reveals that the presence of the HPS nodes among the sampled nodes increases with $\mathrm{P_{HPS}}$. A larger $N_{\mathrm{H}}^{(\mathrm{a})}$ value has two major consequences: (1) a more expensive network to deploy and (2) an energy inefficient

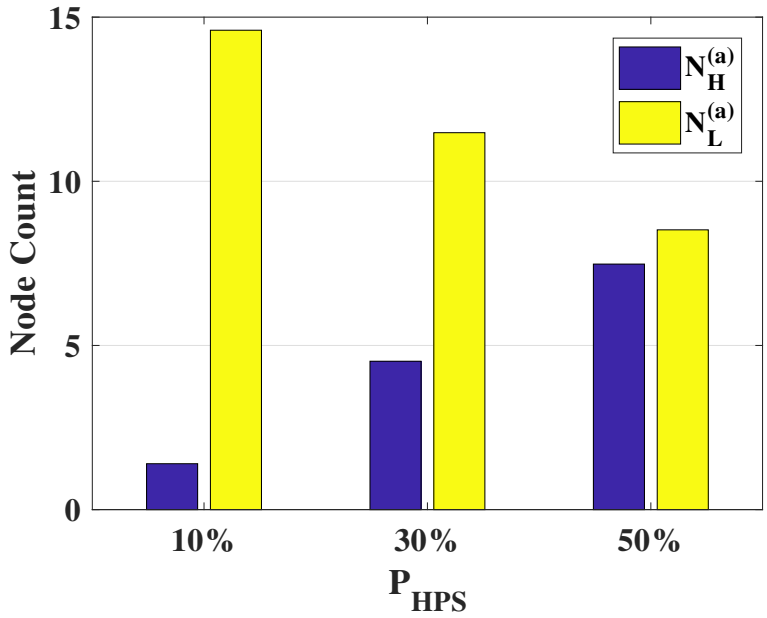


Figure 4.10: Impact of the HPS percentage on the network feasibility. HPS nodes tend to have a higher cost and power consumption. Increasing P_{HPS} from 10% to 50% improves RMSE by only $\sim$ 11%. FP algorithm is used, SNR = 10 dB, $M = 16$.

network due to the relatively higher power consumption of HPS nodes. Moving from $P_{HPS} = 10\%$ to $P_{HPS} = 50\%$ increases $N_H^{(a)}$ by over 4-fold compared to an RMSE improvement of only $\sim$ 11%. Hence, it would be a good trade-off to use a sample with a dominance of LPS measurements.

4.4.5 Experiment 5

In practice the HPS nodes tend to be well maintained and have a stable power source. Hence, it would be reasonable to consider the case where the HPS nodes are always

active while the dynamic scheduling is performed strictly over the LPS nodes. Figure 4.11 compares this scenario with the regular case where the HPS nodes are not necessarily active. Notice that in the case where the HPS nodes are always active - $\text{HPS}_{\text{active}}$- the sample size is an average. This is a result of activating the sleeping HPS nodes after the scheduling is performed. Notice that at low SNR values the $\text{HPS}_{\text{active}}$ case outperforms the dynamic algorithm. This is a result of introducing a set of HPS nodes to a majority of low-quality, noisy LPS nodes. At higher SNR values the dynamic scheduling has a better performance due to the availability of higher-quality, less noisy LPS nodes at better locations. It is also observed that at a higher P_{HPS} value, the differences between the active and dynamic cases are larger. This is a result of introducing a larger number of HPS nodes to the optimized sample. This introduction of nodes after scheduling results in disturbance to the orthogonality between the samples, hence inducing a larger difference gap. For a fair comparison, the sample size M as well as K are matched to a good degree between the two cases.

4.5 Conclusion

This work has investigated the introduction of sparse sensing into a hybrid network consisting of LPS and HPS nodes. The LPS nodes are considered to be wireless and inexpensive, but noisy. The HPS nodes introduce minimal noise, but are expensive and have a relatively higher power consumption. As a case study, the PM10 dataset

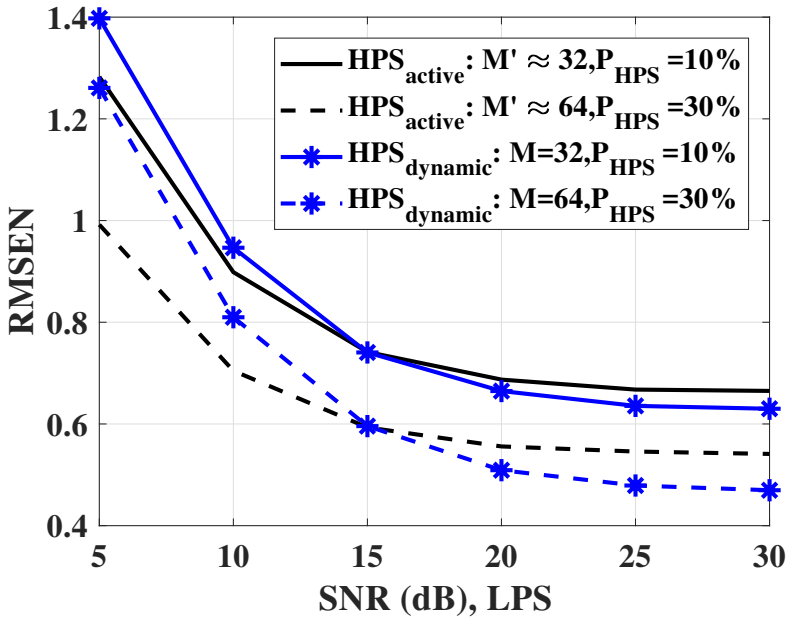


Figure 4.11: Studying the impact of having all the HPS nodes in the active state over the RMSE. M' indicates an average value. The FP algorithm is used for scheduling.

was used as a ground-truth to measure the performance of the presented system. Two major aspects were explored: (1) the impact of using a hybrid network, and (2) the introduction of dynamic scheduling via a greedy algorithm. The use of LPS nodes to augment a sparsely distributed HPS network has proven to reduce the RMS reconstruction error while maintaining a low deployment cost. Moreover, the use of a greedy algorithm to learn from past measurements to adapt the node scheduling has offered a comparable RMSE performance and offered a sampling method that is more resilient to node failure.

Chapter 5

Destination Prediction of Terrain-Aware Mobile Agents Via Inverse Reinforcement Learning

5.1 Introduction

The ability to predict future target's destination and trajectory has numerous practical applications. As we get closer to an era where cars are capable of autonomous driving, and robots are more present in both industrial and domestic settings, the need for surrounding awareness becomes more crucial. For instance, an autonomous

car is expected to avoid any collisions or hindrance of pedestrians or other vehicles on the road [74, 75, 76]. Moreover, a robot is expected to predict the future trajectory of a moving person, and update its own path to avoid any hindrance [77]. In a different sitting, path prediction could be used in defense applications such as missile tracking and interception [78], or for improving target tracking in wireless sensor networks[79].

In a scenario of two players, an agent and a learner, the agent is behaving as a planner trying to solve a decision making problem by searching for the best available action at a given state, while the learner observes the agent with an attempt to learn the factors and preferences impacting the agent's plan. A well known method to model the planning process of an agent is the *Markov decision process* (MDP) [80, 81]. The MDP provides a stochastic framework to optimize for the actions to take at each state in the planning domain. Hence, the agent is trying to solve an MDP problem with an unknown action at a given state. On the other hand, the observer is solving an MDP problem with unknown set of reward effecting the actions taken by the agent—this problem is referred to as *inverse reinforcement learning* (IRL) [82, 83]. If the agent fully knows all the factors that affects its planning, then it is assumed to be perfect planner producing optimized policies —actions taken at a given state— that maximize the cumulative reward values in an MDP problem. Since in real-life scenarios this is usually not the case, the uncertainty in an agent plan need to be accounted for. The concept of maximum entropy applied to the IRL offers a solution to resolve the ambiguities in the actions demonstrated by the agent [77, 84, 85].

Many works in the literature addressed the problem of imitating an agent's behaviour for purposes related to path and destination prediction. Based on the concept of IRL various approaches were presented to imitate an agent's behaviour [77, 85, 86]. relying on a real dataset of taxi-cab trips, the authors in [85] developed a method to learn driver's route preferences and predict their future path and destination. Similarly, the IRL was used to predict the future path of a human actor where in this case a robot was the observer [77]. The robot purpose was to plan a path of its own that would avoid colliding with the predicted human path. In an attempt optimize the reward values from observing a vehicle driving on a highway the IRL algorithm was used [86]. The authors did not employ the maximum entropy in the learning process, resulting in rewards that did not necessarily comply with the agent's. Other works employed different strategies other the IRL framework. A probabilistic approach for pedestrian path prediction is presented in [74]. Rather than using IRL to learn pedestrian preferences, the authors used a form of supervised learning relying on a ground-truth trajectory data set. similar to [74], the authors in [75] used the destination as a hidden variable with a distribution updated using a particle filter, where the intent of the agent at given time was predicted by solving an MDP problem.

Based on a partially observed trajectory this work considers the problem of destination prediction for an agent moving in an open terrain. To the best knowledge of the author, this work is unique form what is seen in related literature in terms of:

† The impact of terrain severity is reflected on the agent trajectory, Section 5.3.

† Agents with different capabilities for terrain traversing are considered.

† The severity of the terrain is utilized as features to learn an agent's behaviour and preferences.

Table 5.1 provides a list of important notations and acronyms used in this work. The rest of this chapter is organized as follows. Section 5.2 presents a concise mathematical background for the IRL problem. A description of the method used to generate the ground-truth data is found in Section 5.3. The set of experiments and their results are discussed in Section 5.5. Finally, conclusion and future work are seen in Section 5.6.

5.2 Formalization

Since the agent is assumed to be solving an MDP problem, the following presents the MDP framework succeeded by that of the IRL to for the purpose of imitation learning.

Table 5.1
Acronyms and Notations

Term	Definition	Term	Definition
$\boldsymbol{s}$	State	$\boldsymbol{S}$	Set of all states
$R(.)$	Reward function	$\boldsymbol{w}$	Feature weights
$\boldsymbol{a}$	Action	$\boldsymbol{A}$	Set of all actions
T	Transition function	π	Policy of an agent
γ	discount factor	$U(\boldsymbol{s})$	Utility of being at a state $\boldsymbol{s}$
$Q(\boldsymbol{s}, \boldsymbol{a})$	Utility of action $\boldsymbol{a}$ being at a state $\boldsymbol{s}$	$\boldsymbol{\xi}$	Agent trajectory
$\boldsymbol{\Xi}$	Set of all trajectories	D_s	Expected state visitation frequency
$\mathfrak{d}$	Agent's destination	$\mathfrak{D}$	Set of all destinations
β	Control parameter	η	Terrain influence parameter
MDP	Markov decision process	IRL	Inverse reinforcement learning
$\boldsymbol{\xi}_{\text{observed}}\%$	Observed percentage of trajectory	∇_{Accuracy}	Maximum change in prediction accuracy

5.2.1 Markov Decision Process

In an MDP problem there are various parameters that govern the plan of an agent. A given planning space is composed of a set of states describing condition of an agent (e.g. position, orientation, velocity). For the work presented in this chapter, a state represent the position of an agent, $\boldsymbol{s} = [x, y]$. The set of all states are expressed as $\boldsymbol{S}$. At each state, there are a set of features describing that state and have the possibility to influence the agent. For instance, a pedestrian would prefer moving on sidewalks and avoiding obstacles like vehicles and other pedestrians. Here, the severity of a terrain is mainly influencing the features in each state $\boldsymbol{f}_s$ —details seen in Section

5.4. Another parameter is the reward function which describes the utility of being at a given state. The reward function is expressed as a weighted sum of the features at a given state [77, 85].

$$R(\boldsymbol{s}) = \boldsymbol{w}^T \boldsymbol{f}_s, \tag{5.1}$$

where $\boldsymbol{w}$ are the wights that expresses the influence of each feature on the value of $R(\boldsymbol{s})$. The reward function can take a negative value, thus inflecting a cost on the agent, or a positive value thereby encouraging being at that state. An agent can transition from on state to the other by taking an action $\boldsymbol{a}$, with $\boldsymbol{A}$ representing the set of all allowable actions. The action considered in this work is the agent's velocity, $\boldsymbol{a} = [v_x, v_y]$. The transition of an agent from one state $\boldsymbol{s}$ to the next $\boldsymbol{s}'$ given an action $\boldsymbol{a}$ is described by the state transition function T. The transition function can be either deterministic $T(\boldsymbol{s}, \boldsymbol{a}) \rightarrow \boldsymbol{s}'$ or stochastic $T(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') \rightarrow P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$. Mapping states to actions is referred to as a policy, π. Since we are considering a stochastic transition function, the policy is also stochastic and gives a distribution over the actions at a given state. The MDP problem is solved by finding the optimum policy π^* which maximizes the agents expected reward as it transitions from one state to the other. The optimal policy can be found by iterating through Bellman's equations. In this work we employ a softened version of the MDP problem to account for the

uncertainty and suboptimality of the agent as a planner [77],

$$\pi^* = \arg \underset{\boldsymbol{a}}{\text{softmax}}\ Q^*(\boldsymbol{s}, \boldsymbol{a}), \tag{5.2}$$

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = R(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}' \in \boldsymbol{S}} T(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') U^*(\boldsymbol{s}'), \tag{5.3}$$

such that $U(\boldsymbol{s})$ is the utility of being at a given state, and where the reward shows the immediate value of being at a state, the utility reflects the long term value of being at that state. $Q(\boldsymbol{s}, \boldsymbol{a})$ is the utility of taking action $\boldsymbol{a}$ after being at state $\boldsymbol{s}$. $\gamma \in (0, 1)$ is the discount factor.

5.2.2 Inverse Reinforcement Learning

In IRL the purpose is to solve an MDP problem where the rewards are unknown based on a set of observation an agent is demonstrating. For this case, the demonstrations are in the form of trajectories $\tilde{\boldsymbol{\xi}}_i = [\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_{M_i}], \tilde{\boldsymbol{\xi}}_i \in \boldsymbol{\Xi}$, where Ξ is the set of all demonstrated trajectories. The method adopted here is based on maximum entropy IRL where the entropy of the distribution over the demonstrated trajectories is maximized [85],

$$\boldsymbol{w}^* = \arg \max_{\boldsymbol{w}}\ \mathcal{L}(\boldsymbol{w}) = \sum_{\tilde{\boldsymbol{\xi}}_i \in \boldsymbol{\Xi}} \log P(\tilde{\boldsymbol{\xi}}_i | \boldsymbol{w}, T), \tag{5.4}$$

$$\equiv \boldsymbol{w}^* = \arg \min_{\boldsymbol{w}}\ \Delta\mathcal{L}(\boldsymbol{w}) \tag{5.5}$$

where $\Delta\mathcal{L}(\boldsymbol{w})$ is the gradient of the log-likelihood function $\mathcal{L}(\boldsymbol{w})$, and can be expressed as [85, 87],

$$\Delta\mathcal{L}(\boldsymbol{w}) = \tilde{\boldsymbol{f}} - \hat{\boldsymbol{f}}_{\boldsymbol{w}}, \tag{5.6}$$

where $\tilde{\boldsymbol{f}} = \frac{1}{K}\sum_i \boldsymbol{f}_{\tilde{\boldsymbol{\xi}}_i}$ is the empirical average of the feature counts $\boldsymbol{f}_{\tilde{\boldsymbol{\xi}}_i}$,

$$\boldsymbol{f}_{\tilde{\boldsymbol{\xi}}_i} = \sum_{\boldsymbol{s}_j \in \tilde{\boldsymbol{\xi}}_i} \boldsymbol{f}_{\boldsymbol{s}_j}.$$

Moreover, $\hat{\boldsymbol{f}}_{\boldsymbol{w}}$ is the expected average of feature counts [85],

$$\hat{\boldsymbol{f}}_{\boldsymbol{w}} = \sum_{\boldsymbol{\xi}} P(\boldsymbol{\xi}|\boldsymbol{w}, T)\boldsymbol{f}_{\boldsymbol{\xi}} \equiv \sum_{\boldsymbol{s}_m \in \boldsymbol{S}} D_{\boldsymbol{s}_m}\boldsymbol{f}_{\boldsymbol{s}_m}, \tag{5.7}$$

where $D_{\boldsymbol{s}}$ is the expected state visitation frequency. Substituting (5.7) back into (5.6),

$$\Delta\mathcal{L}(\boldsymbol{w}) = \tilde{\boldsymbol{f}} - \sum_{\boldsymbol{s}_m \in \boldsymbol{S}} D_{\boldsymbol{s}_m}\boldsymbol{f}_{\boldsymbol{s}_m} \tag{5.8}$$

The minimization in equation (5.5) was carried using the limited-memory BFGS (LBFGS) quasi-newton method.

5.2.3 Destination Inference

Method 1

The distention inference is based on the maximum entropy approach presented in [77]. Given an observed segment of the agent's trajectory $\boldsymbol{\xi}_{1:\mathcal{T}}$, the posterior distribution over the possible destinations can be computed as,

$$P(\mathfrak{d}|\boldsymbol{\xi}_{1:\mathcal{T}}, \boldsymbol{w}) = \frac{P(\boldsymbol{\xi}_{1:\mathcal{T}}|\mathfrak{d}, \boldsymbol{w})P(\mathfrak{d}|\boldsymbol{w})}{P(\boldsymbol{\xi}_{1:\mathcal{T}}|\boldsymbol{w})} = \frac{\frac{e^{R(\boldsymbol{\xi}_{1:\mathcal{T}})+U_{\mathcal{T}+1:\mathfrak{d}}}}{e^{U_{1:\mathfrak{d}}}}P(\mathfrak{d}|\boldsymbol{w})}{\sum_{\substack{\mathfrak{d}'\in\mathfrak{D}\\ \mathfrak{d}'\neq\mathfrak{d}}} \frac{e^{R(\boldsymbol{\xi}_{1:\mathcal{T}})+U_{\mathcal{T}+1:\mathfrak{d}'}}}{e^{U_{1:\mathfrak{d}'}}}P(\mathfrak{d}'|\boldsymbol{w})} \tag{5.9}$$

where $\mathfrak{d} \in \mathfrak{D}$ is the agent's destination, and $\mathfrak{D}$ is the set of all possible destinations. The reward of the observed trajectory is basically the sum of its state rewards, $R(\boldsymbol{\xi}_{1:\mathcal{T}}) = \sum_{\boldsymbol{s}\in\boldsymbol{\xi}_{1:\mathcal{T}}} R(s)$. Using the backward pass algorithm, Algorithm 5, the values of $U_{1:\mathfrak{d}}$ and $U_{\mathcal{T}+1:\mathfrak{d}}$ can be computed by solving an MDP problem with,

† The weights $\boldsymbol{w}$ learned from solving the IRL problem.

† A destination state utility value of zero, while the other states have a large negative utility.

Algorithm 5 Backward Pass

$U'(\boldsymbol{s}) \leftarrow -\infty$
$U'(\mathfrak{d}) = 0$
repeat
  $U(\mathfrak{d}) = U'$
  $U' = \text{softmax}_{\boldsymbol{a}}\ Q^*(\boldsymbol{s}, \boldsymbol{a})$
until $\max |U - U'| \geq \epsilon$
return $P(\boldsymbol{a}_t|\boldsymbol{s}_t, \mathfrak{d}, \boldsymbol{w}) \leftarrow e^{Q^*(\boldsymbol{s},\boldsymbol{a})-U^*(\boldsymbol{s})}$ (*Method 1*)
  $P(\boldsymbol{a}_t|\boldsymbol{s}_t, \mathfrak{d}, \boldsymbol{w}) \leftarrow e^{\beta Q^*(\boldsymbol{s},\boldsymbol{a})}$ (*Method 2*)

Method 2

This method for destination inference is similar to that seen in [88]. The presented algorithm, Algorithm 6, is based on the simple model where an agent has only one destination of interest $\mathfrak{d} \in \mathfrak{D}$. One main difference from method 1 is the use of *Boltzmann policy* to model the distribution of the action taken by the agent, $P(\boldsymbol{a}|\boldsymbol{s}, \mathfrak{d}, \boldsymbol{w}) \equiv e^{\beta Q^*(\boldsymbol{s},\boldsymbol{a})}$. Tweaking the parameter β affects the accuracy of destination prediction. Given an observed portion of its trajectory, the posterior distribution of the agent's destination is expressed as,

$$P(\mathfrak{d}|\boldsymbol{\xi}_{1:\mathcal{T}}, \boldsymbol{w}) \propto P(\boldsymbol{\xi}_{1:\mathcal{T}}|\mathfrak{d}, \boldsymbol{w})P(\mathfrak{d}|\boldsymbol{w}), \tag{5.10}$$

where $P(\mathfrak{d}|\boldsymbol{w})$ is the prior distribution of the destination, and $P(\boldsymbol{\xi}_{1:\mathcal{T}}|\mathfrak{d}, \boldsymbol{w})$ is the probability of the observed trajectory given the destination $\mathfrak{d}$,

$$P(\boldsymbol{\xi}_{1:\mathcal{T}}|\mathfrak{d}, \boldsymbol{w}) = \prod_{t=1}^{\mathcal{T}-1} P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \mathfrak{d}, \boldsymbol{w}). \tag{5.11}$$

The transition probability given a destination $\mathfrak{d}$ is obtained by marginalizing over the action taken at state $\boldsymbol{s}_t$,

$$P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \mathfrak{d}, \boldsymbol{w}) = \sum_{\boldsymbol{a}_t \in A} T(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) P(\boldsymbol{a}_t|\boldsymbol{s}_t, \mathfrak{d}, \boldsymbol{w}). \tag{5.12}$$

Again, using Algorithm 5 the value of $P(\boldsymbol{a}_t|\boldsymbol{s}_t, \mathfrak{d}, \boldsymbol{w})$ can be obtained.

Algorithm 6 Destination Prediction

$j = 1$
for each $\mathfrak{d} \in \mathfrak{D}$ **do**
  $P(\boldsymbol{a}_t|\boldsymbol{s}_t, \mathfrak{d}, \boldsymbol{w}) \leftarrow$ *Backward Pass*
  $P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \mathfrak{d}, \boldsymbol{w})$ (*Equation* (5.12))
  $P(\boldsymbol{\xi}_{1:\mathcal{T}}|\mathfrak{d}, \boldsymbol{w})$ (*Equation* (5.11))
  $P_{\mathfrak{d}}(j) = P(\mathfrak{d}|\boldsymbol{\xi}_{1:\mathcal{T}}, \boldsymbol{w})$ (*Equation* (5.10))
  $j = j + 1$
end for
return $\tilde{\mathfrak{d}} = \arg\max_{\mathfrak{d}} P_{\mathfrak{d}}$

5.3 Ground-Truth Data Generation

In this work, the scenario of interest is that of an agent moving in an $N_x \times N_y$ open field and its trajectory is impacted by the severity of the terrain. The terrain severity is represented by the steepness of a given point in the field, which can be computed by evaluating the gradient magnitude at that point. The method presented in this section is based on a relaxed A-star algorithm, where the terrain severity is incorporated along side distance in the cost function $f(\boldsymbol{s}) = h(\boldsymbol{s}) + g(\boldsymbol{s})$. The proposed

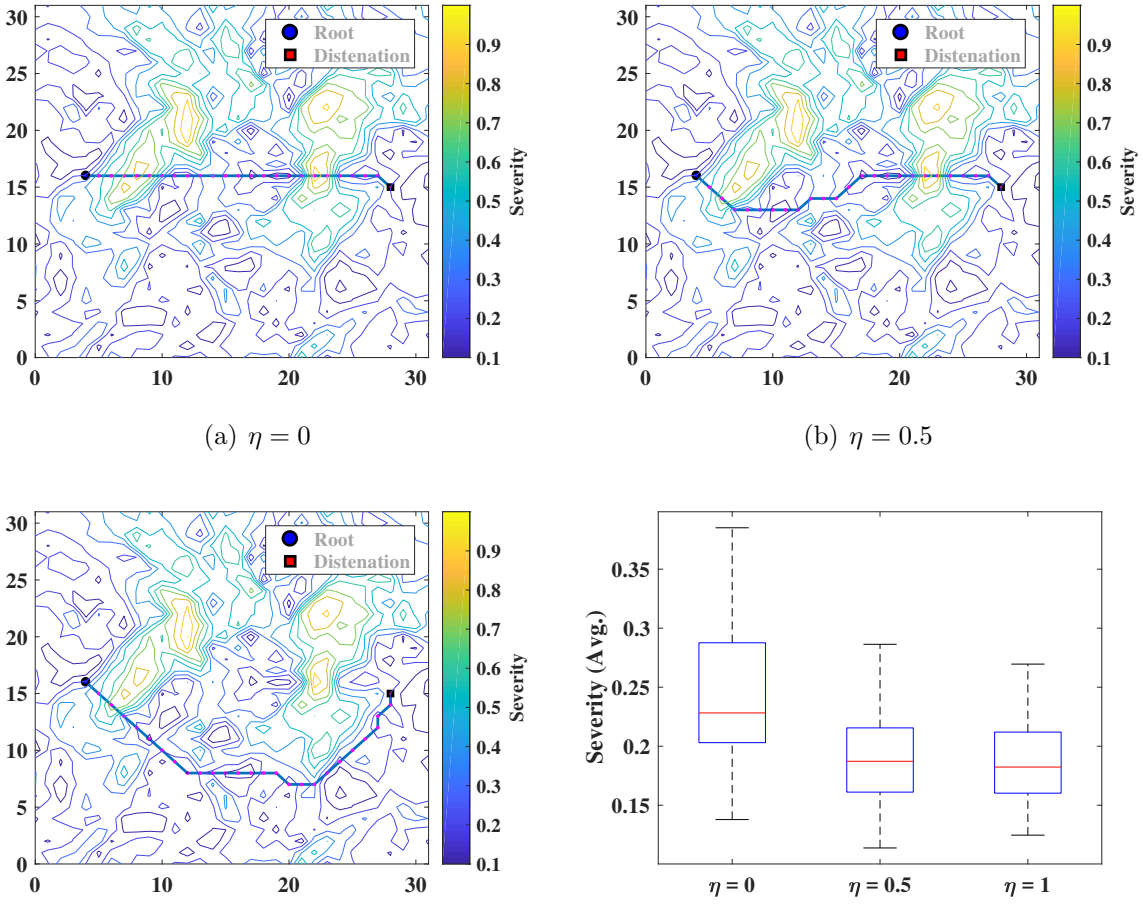
heuristic function is expressed as follows,

$$h(\boldsymbol{s}) = \mathcal{D}(\boldsymbol{s}) + \eta\ \mathcal{G}(\boldsymbol{s}), \tag{5.13}$$

where $\mathcal{D}(\boldsymbol{s})$ is the euclidean distance from state $\boldsymbol{s}$ to the destination, and $\mathcal{G}(\boldsymbol{s})$ is the average severity value on a line connecting point $\boldsymbol{s}$ to the destination. $\eta \in [0, 1]$ is control parameter managing the influence of terrain severity on the generated trajectory. The other part of the cost function $f(\boldsymbol{s})$ is the g-score $g(\boldsymbol{s})$ and it measures the actual cost from the start point to the current point $\boldsymbol{s}$.

$$g(\boldsymbol{s}_t) = g(\boldsymbol{s}_{t-1}) + D_{\boldsymbol{s}_{t-1}\rightarrow\boldsymbol{s}_t} + \eta\ G(\boldsymbol{s}_t), \tag{5.14}$$

where $D_{\boldsymbol{s}_{t-1}\rightarrow\boldsymbol{s}_t}$ is the distance between $\boldsymbol{s}_{t-1}$ and $\boldsymbol{s}_t$. $G(\boldsymbol{s})$ is the gradient magnitude at point n. An optimal trajectory in this scenario would be the shortest with the lowest terrain severity. It is clear that relaxing the admissibility condition —by introducing the severity factor to $h(\boldsymbol{s})$— does not grantee an optimal trajectory. Having said that, the acquired trajectories offered a satisfactory compromise between distance and terrain severity, which was sufficient for the intended purposes in this work, Fig. 5.1. It is also seen how the average severity changes between trajectories generated with different η values, Fig. 5.1(d). Through varying the value of η the generated ground-truth trajectories represent agents with various abilities for traversing the terrain; $\eta = 0$ is the most capable agent, and $\eta = 1$ is the least.



(a) $\eta = 0$ (b) $\eta = 0.5$

(c) $\eta = 1$ (d) Trajectory Statistics

Figure 5.1: Impact of the severity control constant η on the generated trajectory. The higher *eta* is, the more sensitive the algorithm is to terrain severity.

5.4 Feature Generation

The features are based on the minimum distance from a set of per-defined objects in the terrain. Objects are characterized by their normalized severity value $G'(\boldsymbol{s}) \in [0, 1]$. From the trajectory statistics, Fig. 5.1(d), it is seen that on average a trajectory

Figure 5.2: Feature set at state $\boldsymbol{s}$

severity value $G'(\boldsymbol{\xi})$ lies between 0.15 and 0.35. Hence, the objects are categorized into five types as follows,

† Type 1: $0 \leq G'(\boldsymbol{\xi}) < 0.10$

† Type 2: $0.10 \leq G'(\boldsymbol{\xi}) < 0.20$

† Type 3: $0.20 \leq G'(\boldsymbol{\xi}) < 0.30$

† Type 4: $0.30 \leq G'(\boldsymbol{\xi}) < 0.40$

† Type 5: $G'(\boldsymbol{\xi}) \geq 0.40$

This work considers a discretized version of the minimum distance to each object type, with L levels for each type and a total of $5L$ features per state. Figure 5.2 describes the structure of the feature set for a given state $\boldsymbol{s}$, where $f_s^{(l,i)}$ is expressed as,

$$f_s^{(l,i)} = \begin{cases} 1, & D_{min}^{(i)}(\boldsymbol{s}) < l \\ 0, & otherwise, \end{cases} \tag{5.15}$$

where $D_{min}^{(i)}(\boldsymbol{s})$ is the minimum distance between state $\boldsymbol{s}$ and an object of type i.

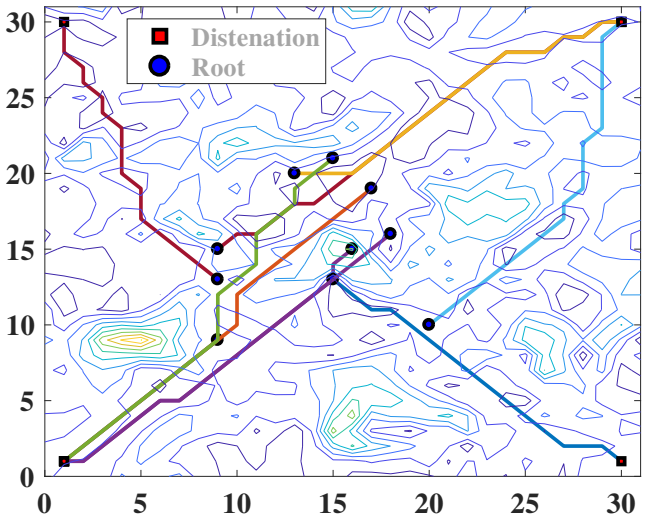


Figure 5.3: Trajectory arrangement for result generation.

5.5 Simulations and Results

Simulations are based on a single randomly generated terrain, hence the results are preliminary and require further iteration are needed. Agents' trajectories are generated such that the starting point is close to the center of the terrain and the destination is one of four goals placed near the corners of the field Fig. 5.3. The trajectory dataset was divided into 70% for training (solving the IRL problem), and the rest are used for testing the prediction accuracy. The accuracy of destination prediction is evaluated as the percentage of correct predictions to the total number of test samples. Table 5.2 gives a short description of the experiments held in this section.

Table 5.2
Experiments Description

	Description
Experiment 1	Study the relationship between the length of the observed portion of the trajectory on the accuracy of prediction. Compare between methods 1 and 2.
Experiment 2	Investigate the terrain impact on the prediction performance.
Experiment 3	Explore the possible gain due to the use of a policy model that is a hybrid between methods 1 and 2.

Experiment 1

In this experiment the performance of both methods 1 and 2 are compared in terms of prediction accuracy. The comparison is held for varying length of the observed portion of the trajectory, Fig. 5.4. According to expectations, as the trajectory observed percentage $\boldsymbol{\xi}_{\text{observed}}\%$ increases, the prediction accuracy improves. The result presented here are for the case where $\eta = 0$, meaning that the agent is very capable of traversing the terrain without hindrance. In the next experiment, we extend the results to include less capable agents.

Experiment 2

The relationship between an agent's capability of traversing a terrain and the accuracy of prediction is investigated. preliminary simulation did not reveal a direct trend

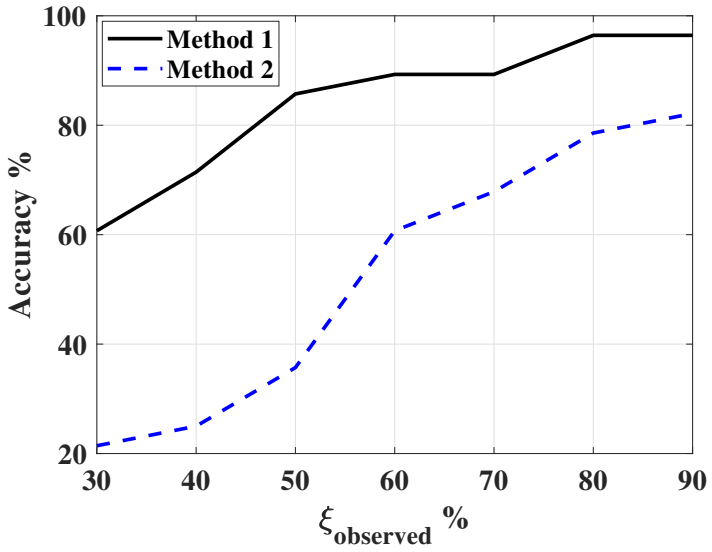
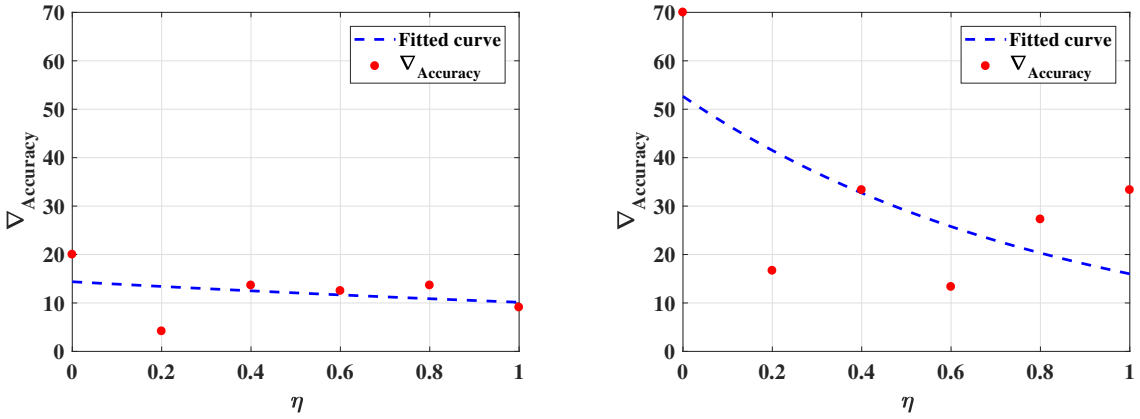


Figure 5.4: Influence of the trajectory's observed portion (as a percentage) over the destination prediction accuracy. Methods 1 and 2 are compared. $\eta = 0$, $\beta = 0.3$.

between η and the prediction accuracy, however it was observed that as the value of η increases, the change in accuracy becomes smaller, Fig. 5.5. The change of accuracy ∇Accuracy% is evaluated as the maximum change of accuracy between any two consecutive values of $\boldsymbol{\xi}_{\text{observed}}$%. A higher value of η represents a less capable agent, where its trajectory would have more maneuvers around severe terrain parts. One possible explanation is that at higher values of η more parts of the terrain are conceived as an obstacle for the agent, and hence the possible variation in its future trajectory is limited. Therefore, the low value of ∇Accuracy% shows that early predictions of an agent's destination are very close to those at higher values of $\boldsymbol{\xi}_{\text{observed}}$%.



(a) Method 1 (b) Method 2

Figure 5.5: Influence of agent's sensitivity to terrain on the maximum change in prediction accuracy. $\beta = 0.3$.

Experiment 3

This experiment introduces a modified model for the learned policy $P(\boldsymbol{a}_t|\boldsymbol{s}_t, \mathfrak{d}, \boldsymbol{w})$ that was mentioned in Algorithm 5. The new policy model for both methods now takes the following format,

$$P(\boldsymbol{a}_t|\boldsymbol{s}_t, \mathfrak{d}, \boldsymbol{w}) = e^{\beta(Q^*(\boldsymbol{s},\boldsymbol{a})-U^*(\boldsymbol{s}))} \tag{5.16}$$

This policy model is related to the energy-based policy model that optimally solves a maximum entropy reinforcement learning problem, [89]. This policy model was only utilized in the prediction phase and not the IRL. Figure 5.6 reveals the impact the use of the new model has on the prediction accuracy. Except for the case of the new

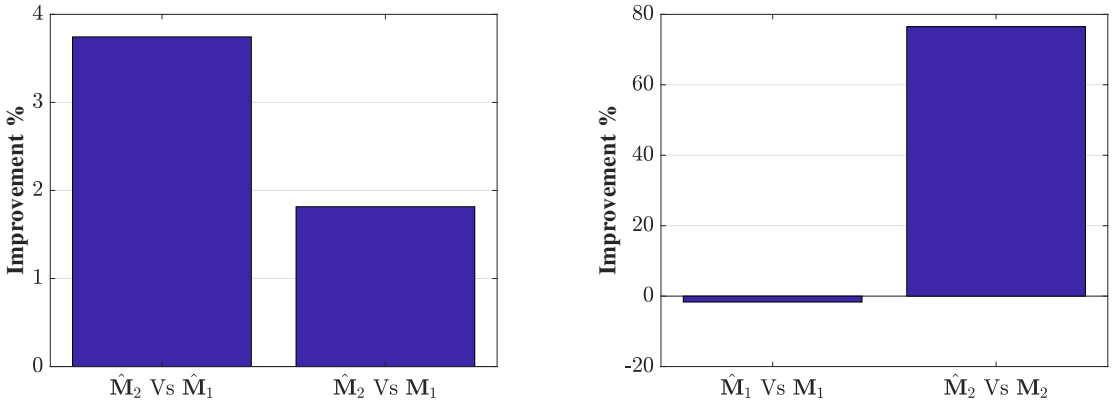


Figure 5.6: The influence the new policy model on the prediction accuracy. Y-axis represents the average accuracy improvement based on $\boldsymbol{\xi}_{\text{observed}}\% = 30\% \rightarrow 90\%$. $\hat{M}_1$ and $\hat{M}_2$ represent methods 1 and 2 with the new policy model. $\beta = 0.3$.

method 1 (as compared to the original), the new policy model offered an improvement on the prediction accuracy. Note that, opposite to what was seen in experiment 1, the second method outperforms first method.

5.6 Conclusion and Future Work

This work discussed the subject of destination predication for an agent moving in a field with varying terrain harshness. The main question under investigation was the impact a terrain has over the prediction accuracy. Two methods were presented and compared in terms of the achieved prediction accuracy. Initial simulations showed the preferable performance of the first method, but that rapidly changed as a new policy

model was introduced to the second method. Simulations also showed that, for both methods, the prediction accuracy does not improve as much with $\boldsymbol{\xi}_{\text{observed}}$ for agents with less capability of traversing harsh terrains. Future work would investigate the use of path and destination prediction to improve target and field coverage in wireless sensor networks.

Chapter 6

Conclusion an Future Work

This work has addressed two of the major challenges encountered in wireless sensor networks, namely, the coverage problem, and the limited power and sensing resources. The coverage problem arises from a poor spatial allocation of the sensor nodes. This is an effect caused by the random deployment of the network forced by the hostility of the environment of interest. On the other hand, a poor resource management system can cause a depletion of power resources and an imprecise monitoring of the phenomena of interest.

The coverage problem was mainly dealt with by relocating the sensor nodes with the goal of achieving a more accurate depiction of the surrounding environment and the event under scrutiny. In Chapter 2, the main interest was the maximization of

the coverage percentage in the field which the network is deployed. Other objectives were also incorporated into the optimization problem to reduce the mobility cost and maintain the connectivity of the network. Due to the NP-completeness of the multi-objective relocation problem, a set evolutionary computation algorithms were considered. While the particle swarm optimization offered a decent coverage percentage at a low computation cost, the artificial immune system algorithm generally offered the best coverage rate. Another form of relocation was considered in Chapter 3 with the purpose of achieving a better tracking accuracy of mobile targets. The relocation was based on analyzing the spatial mobility trend of the targets of interest, and utilizing that to establish a region of interest to carryout the relocation. Multiple criteria were used to optimize the relocation positions inside the ROI. Simulations revealed that the K-coverage criterion offered the best overall performance. However, the simple method of relocating to a uniformly distributed random locations, offered a good compromise between tracking accuracy and computational complexity. In Chapter 5, the accuracy of destination prediction was investigated for targets moving in a terrain with varying levels of severity. targets with different capabilities for traversing a terrain were considered, with the intention of studying the terrain impact on destination prediction. Two methods were presented to infer a target's destination based on an observed portion of the trajectory. As anticipated, simulations showed an improvement in prediction accuracy as more portions of the trajectory are observed.

This improvement was not as obvious in targets with a lower terrain traversing capability. Also, the introduction of an energy-based policy model displayed a clear improvement on the prediction accuracy.

The utilization of sparse sensing for sensor measurement scheduling was presented in Chapter 4. Sparse sensing gives the ability to activate a small set of the available nodes while maintaining enough information to reconstruct the data of the other inactive sensors. This provides the possibility of extending the life-time of the network. Two greedy algorithms along side a simple spatial uniform method were explored to provide the measurement schedule. The frame potential greedy algorithm offered the best balance between reconstruction accuracy and resilience against node failure.

As an extension to the work presented in this dissertation, the following challenges are of interest:

- The use of inverse reinforcement learning for target path prediction with the following goals.
  - Improve the field coverage where targets are expected to be in the future.
  - Enhanced target pursuit and interception capabilities.
  - Study the atmospheric effects on path prediction for airborne targets.
- Incorporating the node residual power along side the information content, to optimize the measurement scheduling. This has the potential of avoiding the

depletion of power resources of nodes that tend to attain higher information content– especially for events of a stationary nature.

References

[1] H. I. Sweidan and T. C. Havens, *Coverage optimization in a terrain-aware wireless sensor network,* 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, 2016, pp. 3687-3694.

[2] Abo-Zahhad M., Ahmed S.M., Sabor N., and Sasaki, S., *Utilization of multi-objective immune deployment algorithm for coverage area maximization with limit mobility in wireless sensors networks,* in Wireless Sensor Systems, IET, vol.5, no.5, pp.250-261, Oct. 2015.

[3] Yourim Yoon, and Yong-Hyuk Kim, *An Efficient Genetic Algorithm for Maximum Coverage Deployment in Wireless Sensor Networks,* in Cybernetics, IEEE Transactions on , vol.43, no.5, pp.1473-1483, Oct. 2013.

[4] Aziz N.A.B.A., Mohemmed A.W., and Alias M.Y., *A wireless sensor network coverage optimization algorithm based on particle swarm optimization and Voronoi*

diagram, in Networking, Sensing and Control, 2009. ICNSC '09. International Conference on , pp.602-607, 26-29 March 2009.

[5] Aziz N.A.A., Mohemmed A.W., Alias M.Y., Aziz, K.A., and Syahali S., *Coverage Maximization and Energy Conservation for Mobile Wireless Sensor Networks: A Two Phase Particle Swarm Optimization Algorithm,* in Bio-Inspired Computing: Theories and Applications (BIC-TA), 2011 Sixth International Conference on , pp.64-69, 27-29 Sept. 2011.

[6] Pradhan P.M., Baghel V., Panda G., and Bernard M., *Energy Efficient Layout for a Wireless Sensor Network using Multi-Objective Particle Swarm Optimization,* in Advance Computing Conference, 2009. IACC 2009. IEEE International , pp.65-70, 6-7 March 2009.

[7] Cheng T.M., and Savkin A.V., *A distributed self-deployment algorithm for the coverage of mobile wireless sensor networks,* in Communications Letters, IEEE , vol.13, no.11, pp.877-879, Nov. 2009.

[8] O. Banimelhem, M. Mowafi and W. Aljoby, "Genetic Algorithm Based Node Deployment in Hybrid Wireless Sensor Networks," Communications and Network, pp. 273-279, vol. 5, no. 4, Nov. 2013.

[9] Zhuofan Liao, Jianxin Wang, Shigeng Zhang, Jiannong Cao, and Geyong Min, *Minimizing Movement for Target Coverage and Network Connectivity in Mobile*

Sensor Networks, in Parallel and Distributed Systems, IEEE Transactions on , vol.26, no.7, pp.1971-1983, July 2015.

[10] Qun Zhao, and Gurusamy M., *Lifetime Maximization for Connected Target Coverage in Wireless Sensor Networks,* in Networking, IEEE/ACM Transactions on , vol.16, no.6, pp.1378-1391, Dec. 2008.

[11] Ramos H.S., Boukerche A., Pazzi R.W., Frery A.C., and Loureiro A.A.F., *Cooperative target tracking in vehicular sensor networks,* in Wireless Communications, IEEE , vol.19, no.5, pp.66-73, Oct. 2012.

[12] Baumgartner K., and Ferrari S., *A Geometric Transversal Approach to Analyzing Track Coverage in Sensor Networks,* in Computers, IEEE Transactions on , vol.57, no.8, pp.1113-1128, Aug. 2008.

[13] B. Wang, *Coverage Problems in Sensor Networks: A Survey,* ACM Computing Surveys, vol. 43, no. 32, Oct 2011.

[14] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* San Francisco, CA, USA: Freeman, 1979.

[15] Topcuoglu H.R., Ermis M., Sifyan M., *Positioning and Utilizing Sensors on a 3-D Terrain Part ITheory and Modeling,* in Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on , vol.41, no.3, pp.376-382, May 2011.

[16] Topcuoglu H.R., Ermis M., Sifyan M., *Positioning and Utilizing Sensors on a 3-D Terrain Part IISolving With a Hybrid Evolutionary Algorithm,* in Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on , vol.41, no.4, pp.470-480, July 2011.

[17] Temel S., Unaldi N., Kaynak O., *On Deployment of Wireless Sensors on 3-D Terrains to Maximize Sensing Coverage by Utilizing Cat Swarm Optimization With Wavelet Transform,* in Systems, Man, and Cybernetics: Systems, IEEE Transactions on , vol.44, no.1, pp.111-120, Jan. 2014.

[18] Zou Y., Krishnendu Chakrabarty, *Sensor deployment and target localization based on virtual forces,* in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies , vol.2, pp.1293-1303, March 30 2003-April 3 2003.

[19] Abo-Zahhad M., Ahmed S., Sabor N., *et al.*, *The convergence speed of single-and multi-objective immune algorithm based optimization problems,* Signal Process., Int. J., vol.4, no.5, pp.247266, 2010.

[20] L. J. Eshelman and J. D. Schaffer, *Real-coded genetic algorithms and interval-schemata,* in Proc. 2nd Workshop Found. Genet. Algorithms, pp. 187202, 1993.

[21] J. Kennedy and R. Eberhart, *Particle swarm optimization,* in Proc. IEEE Int. Conf. Neural Netw., Apr. 1995, pp. 19421948.

[22] Yuhui Shi and Eberhart, R.C., *Empirical study of particle swarm optimization,* in Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on , vol.3, pp.1950, 1999.

[23] Tan, R., Xing, G., Liu, B., Wang, J. and Jia, X., *Exploiting Data Fusion to Improve the Coverage of Wireless Sensor Networks,* in IEEE/ACM Transactions on Networking, vol.20, no.2, pp. 450-462, April 2012.

[24] Wang, G., Cao, G., Berman, P. and La Porta, T. F., *Bidding Protocols for Deploying Mobile Sensors,* in IEEE Transactions on Mobile Computing, vol.6, no.5, pp. 563-576, May 2007.

[25] Kong, L. et al., *Surface Coverage in Sensor Networks,* in IEEE Transactions on Parallel and Distributed Systems, vol.25, no.1, pp. 234-243, Jan. 2014.

[26] Li, J. S. and Kao, H. C., *Distributed K-coverage self-location estimation scheme based on Voronoi diagram,* in IET Communications, vol.4, no.2, pp. 167-177, January 22 2010.

[27] A. N. Njoya et al., *Efficient scalable sensor node placement algorithm for fixed target coverage applications of wireless sensor networks,* in IET Wireless Sensor Systems, vol.7, no.2, pp. 44-54, April 2017.

[28] Garetto M., Gribaudo M., Chiasserini CF. et al., *Sensor deployment and relocation: A unified scheme,* JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY, vol.23, no.3, pp. 400412, May 2008.

[29] Liao Z., Wang, J., Zhang, S., Cao, J. and Min, G., *Minimizing Movement for Target Coverage and Network Connectivity in Mobile Sensor Networks,* in IEEE Transactions on Parallel and Distributed Systems, vol.26, no.7, pp. 1971-1983, July 1 2015.

[30] Guo, Z., Zhou, M. and Jiang, G., *Adaptive Sensor Placement and Boundary Estimation for Monitoring Mass Objects,* in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol.38, no.1, pp. 222-232, Feb. 2008.

[31] Nojeong Heo and P. K. Varshney, *Energy-efficient deployment of Intelligent Mobile sensor networks,* in IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol.35, no.1, pp. 78-92, Jan. 2005.

[32] Olfati-Saber, R. *Distributed Tracking for Mobile Sensor Networks with Information-Driven Mobility,* 2007 American Control Conference, New York, NY, 2007, pp. 4606-4612.

[33] Nguyen, H. T., Ji, Q. and Smeulders, A. W. M., *Spatio-Temporal Context for Robust Multitarget Tracking,* in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.29, no.1, pp. 52-64, Jan. 2007.

[34] Maggio, E. and Cavallaro, A., *Learning Scene Context for Multiple Object Tracking,* in IEEE Transactions on Image Processing, vol.18, no.8, pp. 1873-1884, Aug. 2009.

[35] Kaplan, L. M., *Global node selection for localization in a distributed sensor network,* in IEEE Transactions on Aerospace and Electronic Systems, vol.42, no.1, pp. 113-135, Jan. 2006.

[36] Zoghi, M. and Kahaei, M. H., *Adaptive sensor selection in wireless sensor networks for target tracking,* in IET Signal Processing, vol.4, no.5, pp. 530-536, Oct. 2010.

[37] MUTAMBARA, A.G.O., *Decentralized estimation and control for multisensor systems,* CRC, Boca Raton, FL, 1998.

[38] Yarlagadda, R., Ali, I., Al-Dhahir, N. and Hershey, J., *GPS GDOP Metric,* IEE Proc.-Radar, Sonar Navig., vol.147, no.5, Oct. 2000.

[39] Mao, Yingchi and Yin, Ting, *Sensor Deployment for Mobile Object Tracking in Wireless Sensor Networks,* Advances in Computer Science, Engineering and Applications. Springer Berlin Heidelberg. pp. 637-646. 2012.

[40] Shin S, Park S, Kim Y, Matson ET, *Design and Analysis of Cost-Efficient Sensor Deployment for Tracking Small UAS with Agent-Based Modeling,* Sensors (Basel). vol.16, no.4. April 2016.

[41] Domingo-Perez F., Lazaro-Galilea J. L., Bravo I., Martin-Gorostiza E., Salido-Monzu D., *Sensor deployment for motion trajectory tracking with a genetic algorithm,* Industrial Technology (ICIT), 2015 IEEE International Conference on. March 2015.

[42] Rani, S., Ahmed, S. H. *Multi-hop Routing in Wireless Sensor Networks, An Overview, Taxonomy, and Research Challenges,* (SpringerBriefs in Electrical and Computer Engineering, Springer Singapore, 2016).

[43] Kadar, I., *Optimum geometry selection for sensor fusion,* In Proceedings of SPIE, 3374, pp. 96107. Apr. 1998.

[44] Levanon, N., *Lowest GDOP in 2-D Scenarios,* lEE Proc. Radar, Sonar Navig. vol.147, no.3, pp. 149-155, June 2000.

[45] Foderaro, G., Ferrari, S., and Zavlanos, M., *A Decentralized Kernel Density Estimation Approach to Distributed Robot Path Planning,* 2012.

[46] Sheather, S. J., *Density Estimation,* Statist. Sci. vol.19, no.4, 2004, pp. 588-597. http://projecteuclid.org/euclid.ss/1113832723

[47] Paul Bourke, *Calculating The Area And Centroid Of A Polygon,* July 1988.

[48] Vikas C. Raykar, Ramani Duraiswami and Linda H. Zhao, *Fast Computation of Kernel Estimators,* Journal of Computational and Graphical Statistics, vol.19, no.1, pp. 205-220, March 2010.

[49] A. B. Noel, A. Abdaoui, T. Elfouly, M. H. Ahmed, A. Badawy and M. S. Shehata, *Structural Health Monitoring Using Wireless Sensor Networks: A Comprehensive Survey,* in IEEE Communications Surveys and Tutorials, vol. 19, no. 3, pp. 1403-1423, third quarter 2017.

[50] M. A. Nasirudin, U. N. Za'bah and O. Sidek, *Fresh water real-time monitoring system based on Wireless Sensor Network and GSM,* 2011 IEEE Conference on Open Systems, Langkawi, 2011, pp. 354-357.

[51] S. Bhattacharjee, P. Roy, S. Ghosh, S. Misra, M. S. Obaidat, *Wireless sensor network-based fire detection, alarming, monitoring and prevention system for Bord-and-Pillar coal mines,* Journal of Systems and Software, vol. 85, issue 3, 2012, pp. 571-581.

[52] Jennifer Yick, Biswanath Mukherjee and Dipak Ghosal, *Wireless sensor network survey,* Computer Networks, vol. 52, Issue 12, pp. 2292-2330, 2008.

[53] N. A. Pantazis, S. A. Nikolidakis and D. D. Vergados, *Energy-Efficient Routing Protocols in Wireless Sensor Networks: A Survey,* in IEEE Communications Surveys and Tutorials, vol. 15, no. 2, pp. 551-591, Second Quarter 2013.

[54] A. Wang and A. Chandrakasan, *Energy-efficient DSPs for wireless sensor networks,* in IEEE Signal Processing Magazine, vol. 19, no. 4, pp. 68-78, Jul 2002.

[55] Candes E., *Compressive sampling,* in Proc. Int. Congr. Math., Invited Lectures, pp. 1433-1452, 2006.

[56] Candes E. J., Eldar Y., Needell D. and Randall P., *Compressed sensing with coherent and redundant dictionaries,* Applied and Computational Harmonic Analysis, vol. 31, no. 1, pp. 59-73, May 2010.

[57] Candes E., Romberg J., and Tao T., *Stable signal recovery from incomplete and inaccurate measurements,* Commun. Pure Appl. Math., vol. 59, no. 8, pp. 1207-1223, Aug. 2006.

[58] Nocedal J. and Wright S., *Numerical Optimization,* Springer-Verlag, 1999.

[59] Mallat S. and Zhang Z., *Matching pursuits with time-frequency dictionaries,* IEEE Trans. Signal Processing, vol. 41, no. 12, pp. 3397-3415, Dec. 1993.

[60] Blumensath T. and Davies M., *Iterative hard thresholding for compressive sensing,* Appl. Comput. Harmon. Anal., vol. 27, no. 3, pp. 265-274, 2009.

[61] Zichong Chen, Ranieri J., Runwei Zhang and Vetterli M., *DASS: Distributed Adaptive Sparse Sensing,* Wireless Communications, IEEE Transactions on, vol. 14, no. 5, pp. 2571-2583, May 2015.

[62] Candes E. J. and Wakin M. B., *An introduction to compressive sampling,* IEEE Signal Process. Mag., vol. 25, no. 2, pp. 21-30, Mar. 2008.

[63] EPA Victoria, *Hourly PM10 air monitoring data for 2014.* [Online]. Available: http://www.epa.vic.gov.au/. Retrieved 24 February 2015.

[64] S. Rajasegarar, T. C. Havens, S. Karunasekera, C. Leckie, J. Bezdek, M. Jamriska, A. Gunatilaka, A. Skvortsov and M. Palaniswami, *High-Resolution Monitoring of Atmospheric Pollutants Using a System of Low-Cost Sensors,* Geoscience

and Remote Sensing, IEEE Transactions on, vol. 52, no. 7, pp. 3823-3832, July 2014.

[65] Matheron, Georges, *Principles of geostatistics,* Economic Geology. vol. 58, no. 8, pp 12461266, 1963.

[66] Ford, David, *The Empirical Variogram.* [Online]. Available: faculty.washington.edu/edford. Retrieved 8 March 2018.

[67] G. Quer, R. Masiero, G. Pillonetto, M. Rossi, and M. Zorzi, *Sensing, compression, and recovery for wsns: Sparse signal modeling and monitoring framework,* IEEE Trans. Wireless Commun., vol. 11, no. 10, pp. 34473461, Oct. 2012

[68] Das A. and Kempe D., *Algorithms for subset selection in linear regression,* in Proc. ACM STOC, pp. 45-54, Jul. 2009.

[69] Davis G., Mallat S., and Avellaneda M., *Adaptive greedy approximations,* Constructive Approx., vol. 13, no. 1, pp. 57-98, 1997.

[70] Ranieri J., Chebira A., and Vetterli M., *Near-optimal sensor placement for linear inverse problems,* IEEE Trans. Signal Process., vol. 62, no. 5, pp. 1135-1146, Mar. 2014.

[71] Casazza P. G., Fickus M., Kovacevic J., Leon M., and Tremain J., *A physical interpretation of tight frames,* in Harmonic Analysis and Applications. New York, NY, USA: Springer-Verlag, pp. 51-76, 2006.

[72] J. Ranieri, A. Vincenzi, A. Chebira, *et al.*, *EigenMaps: Algorithms for optimal thermal maps extraction and sensor placement on multicore processors,* DAC Design Automation Conference 2012, San Francisco, CA, 2012, pp. 636-641.

[73] The Audiovisual Communications Laboratory.[Online]. Available: http://rr.epfl.ch/paper/CRZ2015.

[74] E. Rehder and H. Kloeden, *Goal-Directed Pedestrian Prediction,* 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), Santiago, 2015, pp. 139-147.

[75] V. Karasev, A. Ayvaci, B. Heisele and S. Soatto, *Intent-aware long-term prediction of pedestrian motion,* 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, 2016, pp. 2543-2549.

[76] J. Ji, A. Khajepour, W. W. Melek and Y. Huang, *Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control With Multi-constraints,* in IEEE Transactions on Vehicular Technology, vol. 66, no. 2, pp. 952-964, Feb. 2017.

[77] Brian D. Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J. Andrew Bagnell, Martial Hebert, Anind K. Dey, and Siddhartha Srinivasa. *Planning-based prediction for pedestrians,* In Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems (IROS'09). pp. 3931-3936. 2009.

[78] P. V. Hahn, R. A. Frederick and N. Slegers, *Predictive Guidance of a Projectile for Hit-to-Kill Interception,* in IEEE Transactions on Control Systems Technology, vol. 17, no. 4, pp. 745-755, July 2009.

[79] H. I. Sweidan and T. C. Havens, *Sensor relocation for improved target tracking,* IET Wireless Sensor Systems, 2018.

[80] Richard Bellman, *A Markovian Decision Process,* Indiana Univ. Math. J., vol. 6 no. 4, pp. 679684. 1957.

[81] Howard, Ronald A., *Dynamic Programming and Markov Processes,* The M.I.T. Press. 1960.

[82] Russell, S., *Learning agents for uncertain environments (extended abstract),* Proceedings of the Eleventh Annual Conference on Computational Learning Theory. ACM press. 1998.

[83] Ng, A. Y., and Russell, S. J., *Algorithms for inverse reinforcement learning,* In Proceedings of the Seventeenth International Conference on Machine Learning. pp. 663-670. June 2000.

[84] B. D. Ziebart, *Modeling purposeful adaptive behavior with the principle of maximum causal entropy,* PhD thesis, 2010.

[85] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey, *Maximum entropy inverse reinforcement learning,* In Proceedings of the 23rd national

conference on Artificial intelligence (AAAI'08), Anthony Cohn (Ed.), vol. 3. pp. 1433-1438. 2008.

[86] Pieter Abbeel and Andrew Y. Ng., *Apprenticeship learning via inverse reinforcement learning,* In Proceedings of the twenty-first international conference on Machine learning (ICML '04). ACM, New York, NY, USA.

[87] Kris. M. Kitani, Brian. D. Ziebart, James A. Bagnell, and Martial Hebert, *Activity Forecasting*, Computer Vision, ECCV 2012. Springer Berlin Heidelberg. pp. 201-214. 2012.

[88] CL Baker, JB Tenenbaum, RR Saxe, *Goal inference as inverse planning,* Proceedings of the Annual Meeting of the Cognitive Science Society (29). 2007.

[89] T. Haarnoja, H. Tang, P. Abbeel, S. Levine, *Reinforcement Learning with Deep Energy-Based Policies,* ICML 2017.

Appendix A

Let us rewrite $\mathbf{J}_f$ at (3.11) as

$$\mathbf{J}_f = J \begin{bmatrix} \mathbf{V} & \mathbf{U} \\ \mathbf{U}^T & \mathbf{D} \end{bmatrix},$$

such that $\mathbf{V} = [\mathbf{J}_p]_{1:2,1:2} + \mathbf{J}$, $\mathbf{U} = [\mathbf{J}_p]_{1:2,3:4}$ and $\mathbf{D} = [\mathbf{J}_p]_{3:4,3:4}$. Taking the inverse of $\mathbf{J}_f$,

$$\mathbf{J}_f^{-1} = \begin{bmatrix} \left(\mathbf{V} - \mathbf{U}\mathbf{D}^{-1}\mathbf{U}^T\right)^{-1} & -\mathbf{U}\mathbf{V}^{-1}\left(\mathbf{D} - \mathbf{U}\mathbf{V}^{-1}\mathbf{U}^T\right)^{-1} \\ -\mathbf{D}^{-1}\mathbf{U}^T\left(\mathbf{V} - \mathbf{U}\mathbf{D}^{-1}\mathbf{U}^T\right)^{-1} & \left(\mathbf{D} - \mathbf{U}\mathbf{V}^{-1}\mathbf{U}^T\right)^{-1} \end{bmatrix}. \tag{A.1}$$

Thus, the MS position error at (3.12) can be expressed as

$$\rho(\mathcal{S}_a) = \mathrm{tr}\{\left(\mathbf{V} - \mathbf{U}\mathbf{D}^{-1}\mathbf{U}^T\right)^{-1}\},$$

which can be expanded as

$$\rho(\mathcal{S}_a) = \frac{\mathrm{tr}\{\mathbf{V} - \mathbf{U}\mathbf{D}^{-1}\mathbf{U}^T\}}{\det\{\mathbf{V} - \mathbf{U}\mathbf{D}^{-1}\mathbf{U}^T\}}. \tag{A.2}$$

Now, let us reformulate $\mathbf{V} - \mathbf{U}\mathbf{D}^{-1}\mathbf{U}^T$,

$$\begin{aligned}\mathbf{V} - \mathbf{U}\mathbf{D}^{-1}\mathbf{U}^T =& \mathbf{J} + [\mathbf{J}_p]_{1:2,1:2} - \\ & - [\mathbf{J}_p]_{1:2,3:4}[\mathbf{J}_p]^{-1}_{3:4,3:4}[\mathbf{J}_p]^T_{1:2,3:4} \\ =& \mathbf{J} + \tilde{\mathbf{J}}_p. \end{aligned} \tag{A.3}$$

Applying (A.3) to (A.2) results in

$$\begin{aligned}\mathrm{tr}\{\mathbf{J} + \tilde{\mathbf{J}}_p\} &= \mathrm{tr}\{\mathbf{J}\} + \mathrm{tr}\{\tilde{\mathbf{J}}_p\} \\ \det\{\mathbf{J} + \tilde{\mathbf{J}}_p\} &= \det\{\mathbf{J}\} + \det\{\mathbf{J}_p\} \\ &\quad + [\mathbf{J}]_{1,1}[\tilde{\mathbf{J}}_p]_{2,2} + [\mathbf{J}]_{2,2}[\tilde{\mathbf{J}}_p]_{1,1} - 2[\mathbf{J}]_{1,2}[\tilde{\mathbf{J}}_p]_{2,1},\end{aligned}$$

which gives the expression at (3.13).

Appendix B

The uncertainty in the first phase comes from two sources: 1) state estimation error, and 2) measurement error, refer to equations (3.6) and (3.8). The KDE in the second phase is affected by position estimation uncertainty found in the state covarience matrix P. The KDE is expressed as,

$$p(\mathbf{X}, \nu) = \frac{1}{n\nu} \sum_{i=1}^{n} K\left(\frac{\mathbf{X} - \mathbf{X}_i}{\nu}\right), \tag{B.1}$$

In our work the Gaussian radial basis function is chosen as the kernel $K(.)$. Hence, equation (B.1) is a sum of non-linear functions $K_i(\mathbf{X}) = K\left(\frac{\mathbf{X}-\mathbf{X}_i}{\nu}\right)$, the resulting uncertainty can be represented as,

$$\Sigma^p = \boldsymbol{J}\Sigma^{\mathcal{X}}\boldsymbol{J}^T, \tag{B.2}$$

where $\boldsymbol{J}$ is the Jacobian matrix of $p(\mathbf{X}, \nu)$. Also, $\mathcal{X} = [\boldsymbol{X_1}, \boldsymbol{X_2}, ..., \boldsymbol{X_n}]$. The uncertainty matrix $\Sigma^{\mathcal{X}}$ is expressed as,

$$\Sigma^{\mathcal{X}} = \begin{bmatrix} \Sigma_{\boldsymbol{X_1}} & \Sigma_{\boldsymbol{X_1 X_2}} & ... & \Sigma_{\boldsymbol{X_1 X_n}} \\ \Sigma_{\boldsymbol{X_2 X_1}} & \Sigma_{\boldsymbol{X_2}} & ... & \Sigma_{\boldsymbol{X_2 X_n}} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{\boldsymbol{X_n X_1}} & \Sigma_{\boldsymbol{X_n X_2}} & ... & \Sigma_{\boldsymbol{X_n}} \end{bmatrix} \tag{B.3}$$

where $\Sigma_{\boldsymbol{X_i}} = [P^{(i)}]_{1:2,1:2}$ is the variance in the i'th estimated location, and $\Sigma_{\boldsymbol{X_i X_j}} = cov(\boldsymbol{X}_i, \boldsymbol{X}_j)$. In the case of $\boldsymbol{X}_i$'s are independent from each other, $\Sigma^{\mathcal{X}}$ reduces to a diagonal matrix. The third phase is affected by the uncertainty from the previous phases through the evaluation of the region of interest, which is the contour/boundary of $p(\mathbf{X}, \nu)$. Hence, the uncertainty is directly related to the one seen in equation (B.2).

Appendix C

This appendix presents a time complexity analysis of the proposed algorithm. It is important to notice that the second and third phases of the proposed system are mainly concerned about the relocation of the nodes. The relocation processes is based on a well established mobility trend, which is the result of tracking targets for a span of time using the initial deployment. Since this process does not occur frequently, its computational complexity is not critical for an online system operation. Hence, the dominant operation would lie in the target tracking (phase I). Having said that, the following presents in some detail the time complexity analysis for the three phases.

C.1 Tracking (Phase I)

The tracking algorithm has two procedures, one is for selecting the nodes responsible for estimating the target state (GNS algorithm), while the other performs the

estimation (DEIF algorithm). Let's start with later.

C.1.1 Complexity Analysis (DEIF)

The following is an evaluation of algorithm 7 time complexity at a given sensing node,

Algorithm 7 DEIF

procedure PREDICTION
2: $\bar{\mathbf{s}}_k = \mathbf{G}\hat{\mathbf{s}}_{k-1}$
$\bar{\mathbf{P}}_k = \mathbf{G}\mathbf{P}_{k-1}\mathbf{G}^T + \sigma_u^2\mathbf{B}\mathbf{B}^T$
4: **end procedure**
procedure CORRECTION
6: $\hat{\mathbf{s}}_k = \mathbf{P}_k\left(\bar{\mathbf{P}}_k^{-1}\bar{\mathbf{s}}_k + \sum_{j\in\mathcal{S}_a}\boldsymbol{\eta}_k^{(j)}\right)$
$\mathbf{P}_k = \text{inv}\left(\bar{\mathbf{P}}_k^{-1} + \sum_{j\in\mathcal{S}_a}\boldsymbol{\Omega}_k^{(j)}\right)$
8: **end procedure**
return $\hat{s}_k, \mathbf{P}_k$

† Line 2: G is a $n \times n$ matrix and $\hat{s}_{k-1}$ is a $n \times 1$ vector. For one multiplication operation the time complexity is $O(n^2)$. Here n is the length of the state vector.

† Line 3: P_{k-1} is a $n\times n$ matrix, and B is a $n\times m$ matrix, such that m is the number of measurements. For the first term in the summation there is one transpose operation with $O(n^2)$, and two matrix multiplications each with $O(n^3)$, hence $O(n^2+2n^3)$. The second term has two multiplications and one transpose, hence $O(n^2 + mn^2 + mn)$. The summation accounts for $O(n^2)$. Therefore the total complexity is $O(2n^3 + 3n^2 + mn^2 + mn)$.

† Line 6: The first term in the parentheses gives $O(n^3+n^2)$. The second term has

a complexity of $O(ln)$, where $l = |\mathcal{S}_a|$ is the number of active nodes. Adding the two terms requires $O(n)$. For the outer multiplication $O(n^2)$. Hence the complexity here is $O(n^3 + 2n^2 + ln + n)$.

† Line 7: The inverse of $\bar{\mathbf{P}}_k$ has been already computed in line 6. The complexity of the operation inside the parentheses is $O(ln^2 + n^2)$, where $\Omega_k^{(j)}$ is a $n \times n$ matrix. The outer inverse operation requires $O(n^3)$. The complexity in this line is $O(n^3 + ln^2 + n^2)$.

Since the DEIF algorithm is recursive, the complexity of a single run represents the algorithm complexity. From the previous analysis the total complexity is $O(4n^3 + 7n^2 + mn^2 + ln^2 + ln + mn + n)$ which simplifies to $O(n^3 + ln^2 + mn^2)$. In our work $m < n$, hence the complexity further simplifies to $\boldsymbol{O(n^3 + ln^2)}$.

C.1.2 Complexity Analysis (GNS)

The algorithm for node selection is based on the work presented in [35], where similar to our work the GNS algorithm was paired with a DEIF for target tracking. Even though that work has not provided a time complexity analysis of the GNS algorithm, it did perform an experiment using a real dataset for a targets moving at an average speed of 20 m/s. Based on our work, Table C.1 shows the time taken for a single run of the tracking algorithm, which includes performing node selection via the GNS

Table C.1
Execution Time of a Single Run of the Tracking Algorithm

$\boldsymbol{R_{GNS}}$	**Avg.** $\lvert\mathcal{S}_a\rvert$	**Avg. Execution Time (sec.)**
10	0.7	0.001
15	1.3	0.0013
20	1.9	0.0017
25	2.4	0.0023
30	2.8	0.0032

algorithm and target location estimation via the DEIF algorithm. For $\boldsymbol{R_{GNS}} = 30$, around $3ms$ is required, so for a target moving at a 20 m/s it would only have moved $6cm$, which would be fairly reasonable for online tracking.

C.2 KDE (Phase II)

In this phase, a density function is fitted to a data-set of estimated target locations using the KDE with the purpose of establishing the region of interest. Depending on the algorithm used for computing the KDE, its complexity can range from a quadratic $O(\ell\kappa)$ evaluations of the kernel function $K(.)$, to a linear $O(\ell+\kappa)$ number of evaluations, where ℓ is the number of sample points and κ is the number of evaluation points [48]. In our work we used the "*ksdensity*" Matlab function, for which there is no indication of using an algorithm that would enhance its computational complexity. Hence, we assume the worst case scenario of a direct implementation of the KDE with $\boldsymbol{O(\ell\kappa)}$ evaluations.

C.3 Relocation (Phase III)

Here, the relocation is done by finding a set of new sensor locations that would optimize a given objective function. The Genetic Algorithm is utilized for this purpose. The GA is a probabilistic optimization technique, which means that it is not trivial to evaluate a theoretical expression that describes its time complexity for different problems. Having said that, one might roughly express this complexity based on the number of evaluations of the fitness/objective function. Hence, for a fixed number of generations and population size, the time complexity is in the order of $O(N_{\text{pop}} g_{\max})$ evaluations. Table C.2 provides the time complexity for each of the fitness functions used in this work. The following gives some details on what is presented in Table C.2.

† The GDOP fitness function as seen in equation (3.20) requires N_{VTP} evaluations of (3.19). The time complexity of (3.19) is $O(d^3 + ld^2)$, where d is the dimensionality of a grid point and l is the number of active nodes. Hence, equation (3.20) requires $O(d^3 N_{\text{VTP}} + ld^2 N_{\text{VTP}})$. In this work $l \leq 3$, $d = 2$ and $N_{\text{VTP}} > l$, therefore the time complexity can reduce to $\approx O(N_{\text{VTP}})$.

† To compute the K-Coverage of a single VTP, we need to find the number of sensors within a distance R_s from that VTP, which requires $O(|\mathcal{S}_{\text{final}}|)$. The K-Coverage fitness is the average over N_{VTP} points, hence $O(N_{\text{VTP}}|\mathcal{S}_{\text{final}}|)$.

Table C.2
Computational Complexity of Used Fitness Functions

	Complexity
GDOP	$\approx O(N_{\text{VTP}})$
K-Coverage	$O(N_{\text{VTP}}\|\mathcal{S}_{\text{final}}\|)$
Coverage	$O(\|\mathcal{G}\|\|\mathcal{S}\|)$
Distance	$O(\|\mathcal{S}_{mob}\|^2)$

† The coverage fitness in equation (3.22) has a computational complexity that is governed by A_{cov}, which in turn requires $O(|\mathcal{G}||\mathcal{S}|)$.

† The mobility cost depends on computing (3.24), which can be easily shown to have a complexity of $O(|\mathcal{S}_{mob}|^2)$.

One final note to mention is that for the evaluation of both the GDOP and K-Coverage fitness functions, it is necessary to have the distance between each VTP and the sensor nodes. This distance is used to select the closest M sensors for evaluating the fitness. This can produce an overhead of $O(N_{\text{VTP}}|\mathcal{S}_{\text{final}}|)$ if computed along side the fitness function. While this can be alleviated by computing these distances a head of time, in this work the distances where computed as a part of the fitness evaluation.

C.4 Execution Time (Phase II and Phase III)

Here we present the execution times for both phase II and phase III. The average execution time for phase II (KDE algorithm) is 2 seconds. As for phase III (relocation),

Table C.3
Execution Time for the Relocation Algorithm Using Different Objective Functions

	Avg. Execution Time (sec.)
GDOP	135.6
K-Coverage	5.8
Coverage†	60.7

† Coverage optimization outside the ROI (if used its exc. time is added to that of GDOP and K-Coverage).

the execution time for the different objective functions is presented in Table C.3. It is clear that the optimization for the relocation process requires more time than that of tracking. For our intended purpose, this would be reasonable since the relocation processes is not an online operation.

Appendix D

Letters of Permission

Title: Coverage optimization in a terrain-aware wireless sensor network
Conference Proceedings: Evolutionary Computation (CEC), 2016 IEEE Congress on
Author: Husam I. Sweidan
Publisher: IEEE
Date: July 2016

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

The Institution of Engineering and Technology
Michael Faraday House
Six Hills Way, Stevenage
Hertfordshire, SG1 2AY
United Kingdom
T +44 (0)1438 313311
F +44 (0)1438 765526
www.theiet.org

Ref: HIS/Permission/WSS.0037

FAO: Husam Sweidan,
411 MacInnes Dr.,
Houghto, MI, 49931
USA.
+1 (906) 281-8188
eng.husam86@gmail.com

12 April 2018

Permission to reproduce IET content

Dear Sirs,

Sensor relocation for improved target tracking ("the Material") to be used in the thesis 'Resource Optimization in Wireless Sensor Networks for An Improved Field Coverage and Cooperative Target Tracking', by Husam Sweidan, to be published by Michigan Technological University ("the Work")

The Institution of Engineering and Technology ("the IET") hereby grants to Husam Sweidan ("HIS") non-exclusive permission to use the Material in the Work subject to the terms set out below:

Territory:	Worldwide
Languages:	All languages
Term:	Life of the Work
Media:	All print and electronic media how known or hereafter devised
Credit:	Sweidan, H.I., Havens, T.C.: 'Sensor relocation for improved target tracking', IET Wireless Sensor Systems, 2018, 8, (2), pp. 76-86
Fee:	None

The permission granted by this letter may not be licensed or assigned without the prior written consent of the IET.

Neither party shall be liable to the other for indirect, incidental, special or consequential damages arising out of or in relation to this agreement (unless such liability cannot be excluded or limited by law).

The parties agree that this letter constitutes the entire agreement between them relating to the use of the Material by HIS and supersedes all previous agreements, understandings and arrangements between them, whether in writing or oral in respect of its subject matter.

No variation of this agreement shall be valid or effective unless it is in writing, refers to this agreement and is duly signed or executed by, or on behalf of, each party.

This letter and any dispute or claim arising out of, or in connection with, it, its subject matter or formation (including non-contractual disputes or claims) shall be governed by, and construed in accordance with, the laws of England and Wales and the parties irrevocably agree that the courts of England and Wales shall have exclusive jurisdiction to settle any

dispute or claim arising out of, or in connection with, this letter, its subject matter or formation (including non-contractual disputes or claims).

Please sign and return the enclosed copy of this letter to confirm your agreement to it.

Yours faithfully

[redacted]

James Sutherland
Permissions Officer

We confirm our agreement to the terms set out above

Signed: [redacted]
Print name: Husam Swedan

Date: 04-A-2018